



PHP & MySQL

Dynamische Websites
entwickeln

JON DUCKETT

JON DUCKETT

PHP & MySQL

Übersetzung aus dem Amerikanischen
von Isolde Kommer

WILEY
WILEY-VCH GmbH

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

1. Auflage 2022

© 2022 Wiley-VCH GmbH, Boschstr. 12, 69469 Weinheim, Germany

Original English language edition PHP & MySQL © 2022 by Wiley Publishing, Inc.
All rights reserved including the right of reproduction in whole or in part in any form. This translation published by arrangement with John Wiley and Sons, Inc.

Copyright der englischsprachigen Originalausgabe PHP & MySQL © 2022 by Wiley Publishing, Inc.
Alle Rechte vorbehalten inklusive des Rechtes auf Reproduktion im Ganzen oder in Teilen und in jeglicher Form. Diese Übersetzung wird mit Genehmigung von John Wiley and Sons, Inc. publiziert.

Wiley, the Wiley logo, and related trademarks and trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries. Used by permission.

JavaScript and MySQL are trademarks of Oracle America, Inc.

Wiley und darauf bezogene Gestaltungen sind Marken oder eingetragene Marken von John Wiley & Sons, Inc., USA, Deutschland und in anderen Ländern.

Das vorliegende Werk wurde sorgfältig erarbeitet. Dennoch übernehmen Autoren und Verlag für die Richtigkeit von Angaben, Hinweisen und Ratschlägen sowie eventuelle Druckfehler keine Haftung.

Cover: Susan Bauer, Heppenheim

Korrektur: Claudia Lötschert

Satz: Isolde Kommer

Druck

CPI Group (UK) Ltd, Croydon CR0 4YY

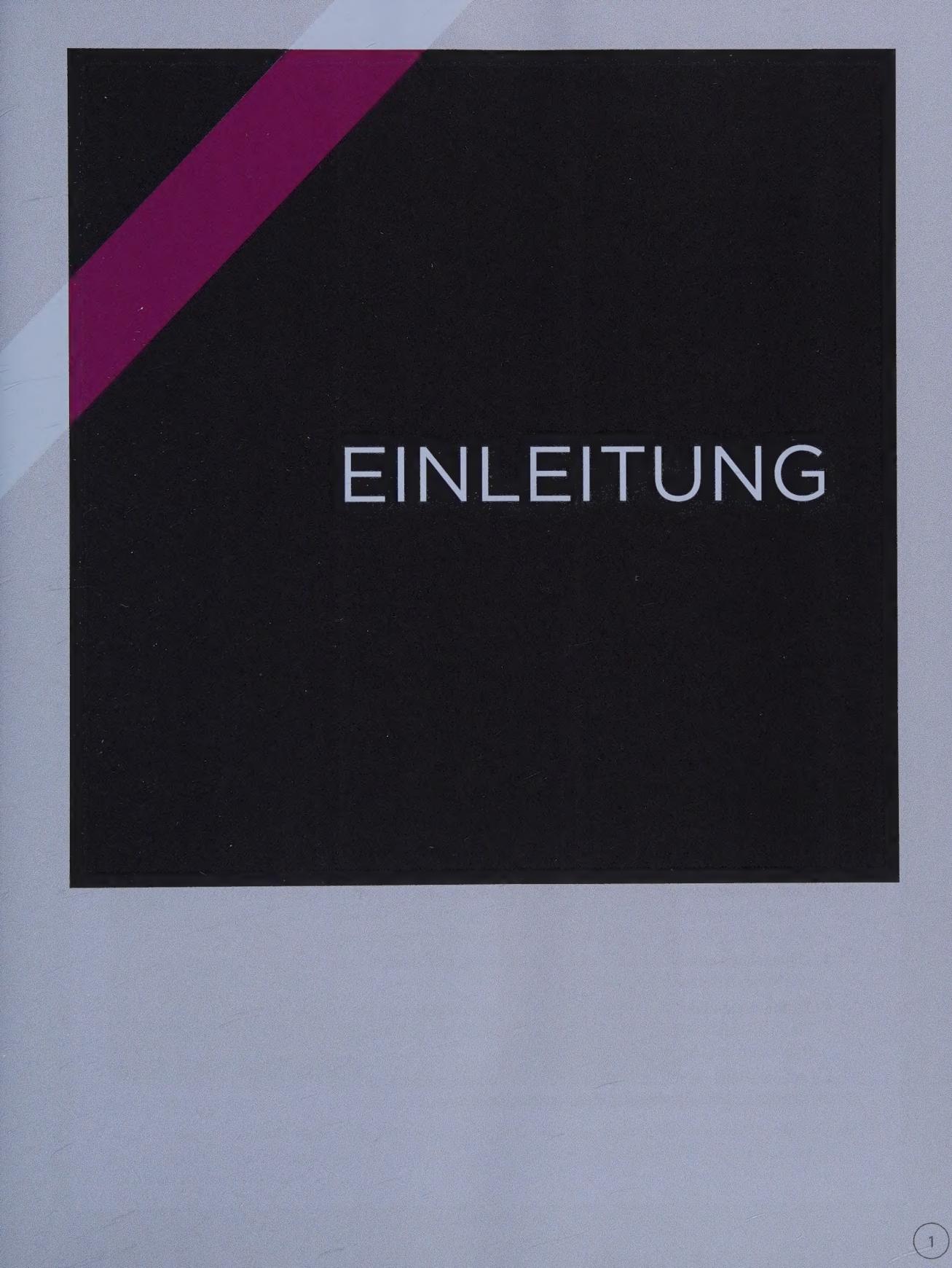
Print ISBN: 978-3-527-76070-1

C085140-150322

INHALTSVERZEICHNIS

Einleitung	1
Teil A	Grundlagen der Programmierung	17
Kapitel 1	Variablen, Ausdrücke & Operatoren	29
Kapitel 2	Kontrollstrukturen.....	67
Kapitel 3	Funktionen.....	103
Kapitel 4	Objekte & Klassen.....	143
Teil B	Dynamische Webseiten	177
Kapitel 5	Eingebaute Funktionen	201
Kapitel 6	Daten von Browsern erhalten.....	231
Kapitel 7	Bilder & Dateien	285
Kapitel 8	Datum & Uhrzeit.....	309
Kapitel 9	Cookies & Sitzungen	329
Kapitel 10	Fehlerbehandlung	349
Teil C	Datenbankgestützte Websites	381
Kapitel 11	Structured Query Language	397
Kapitel 12	Daten aus der Datenbank abrufen.....	433
Kapitel 13	Daten in der Datenbank aktualisieren.....	483
Teil D	Die Beispielanwendung ausbauen	521
Kapitel 14	Refactoring & Dependency Injection.....	533
Kapitel 15	Namensräume & Bibliotheken.....	557
Kapitel 16	Mitgliedschaft	603
Kapitel 17	Weitere Funktionen hinzufügen.....	633
Index	662

Code-Download: <http://phpandmysql.com>



EINLEITUNG

In diesem Buch lernen Sie, Websites mit der Programmiersprache PHP zu erstellen und die Daten für die Website in einer Datenbank wie MySQL zu speichern.

PHP ist eine Programmiersprache, die für die Ausführung auf einem Webserver konzipiert wurde. Wenn jemand eine Webseite anfordert, kann der Server mithilfe von PHP eine HTML-Seite generieren, die dann an den jeweiligen Benutzer zurückgeschickt wird. Dies bedeutet, dass die Seiten auf einzelne Personen zugeschnitten werden können. Das ist erforderlich für Websites, auf denen die Benutzer beispielsweise die folgenden Aufgaben ausführen können:

- **Sich registrieren oder einloggen**, denn der Name, die E-Mail-Adresse und das Passwort sind von Benutzerin zu Benutzer unterschiedlich.
- **Einen Kauf abschließen**, denn die Bestell-, Zahlungs- und Lieferdaten eines jeden Kunden sind einzigartig.
- **Die Website durchsuchen**, denn die Suchergebnisse sind individuell auf jeden Nutzer zugeschnitten.

PHP wurde entwickelt, um in Verbindung mit Datenbanken wie MySQL eingesetzt zu werden, in denen Daten abgelegt werden können, wie zum Beispiel der Inhalt der angezeigten Seiten, die Produkte, die auf einer Webseite zum Kauf angeboten werden, oder Details über die registrierten Benutzer der Webseite. Sie lernen, wie Sie mithilfe von PHP Webseiten erstellen können, auf denen registrierte Benutzer dann die in der Datenbank gespeicherten Daten aktualisieren können. Zum Beispiel:

- **Content-Management-Systeme** ermöglichen es den Verantwortlichen, den Inhalt der Webseite per Formular zu aktualisieren. Diese Aktualisierungen werden dann auf der Website präsentiert, ohne dass dafür neuer Code geschrieben werden müsste.
- **Onlineshops** ermöglichen es den Seitenbetreibern, ihre Produkte zum Verkauf anzubieten, und den Kunden, Einkäufe zu tätigen.
- **Soziale Netzwerke** bieten die Möglichkeit, sich zu registrieren und einzuloggen, Benutzerprofile zu erstellen, eigene Inhalte hochzuladen und Seiten zu betrachten, die auf die individuellen Interessen des Nutzers zugeschnitten sind.

Da auf derartigen Websites angezeigte Informationen in einer Datenbank hinterlegt sind, werden diese auch als **datenbankgestützte Websites** bezeichnet.

Hier finden Sie eine Übersicht über die verschiedenen Seiten-typen in diesem Buch. Verschie-dene Seitengestaltungen vermit-teln unterschiedliche Inhalte.

VERGLEICHSOPERATOREN VERWENDEN

LOGISCHE OPERATOREN VERWENDEN

CODE-SEITEN

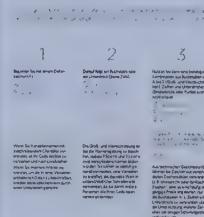
Auf beigefarbenem Hintergrund erfahren Sie, wie Sie einzelne Code-Abschnitte anwenden können.



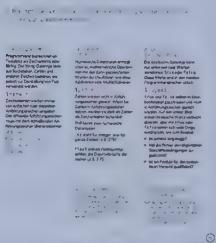
BEISPIELSEITEN

In den vorderen Kapiteln treffen Sie auf diese Seiten, die die gelernten Themen und deren Anwendung nochmals zusammenfassen.

ZUR BENENNUNG VON VARIABLEN



SKALARE (EINFACHE) DATENTYPEN



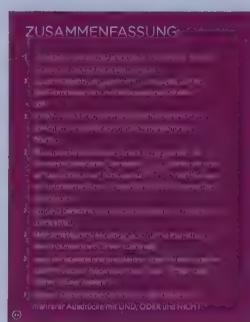
INFORMATIONSSEITEN

Auf weißem Hintergrund führen sie in bestimmte Themen ein und erläutern Zusammenhänge und Funktionsweisen.



DIAGRAMMSEITEN

Diese dunkel hinterlegten Seiten erklären Konzepte anhand von Diagrammen und Infografiken.



ZUSAMMENFASSUNGSSEITEN

Diese erscheinen am Ende eines jeden Kapitels und rufen Ihnen die Hauptinhalte nochmals in Erinnerung.

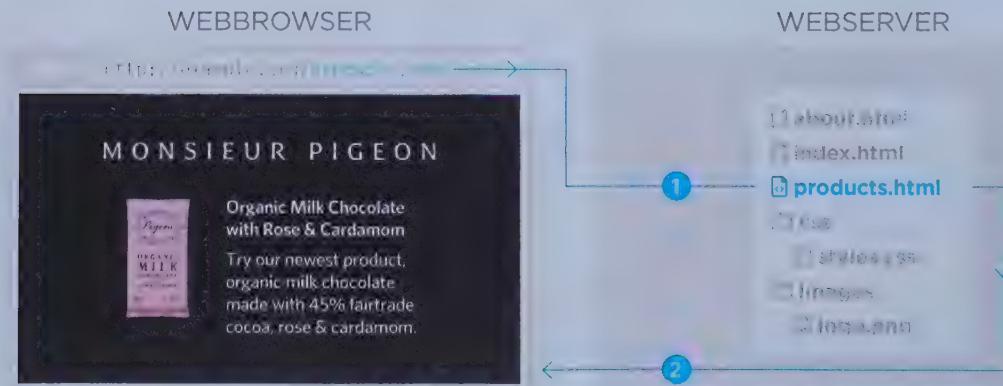
STATISCHE UND DYNAMISCHE WEBSITES IM VERGLEICH

Websites, die nur auf HTML und CSS basieren, zeigen jedem Besucher denselben Inhalt, weil der Server immer dieselben HTML- und CSS-Dateien übermittelt.

1. Wenn Ihr Browser eine Seite einer mit HTML- und CSS-Dateien erstellten Website anfordert, geht die Anfrage an einen **Webserver**, der die Website hostet.
2. Der Webserver ruft die vom Browser angeforderte HTML-Datei ab und sendet sie an den Browser zurück. Möglicherweise überträgt er auch CSS zur Ausgestaltung der Seite, Medien (z. B. Bilder), JavaScript und andere von der Seite verwendete Dateien.

Da alle Besucher die gleichen HTML-Dateien bekommen, sehen sie auch alle die gleichen Inhalte. Deswegen wird diese Art von Website auch als **statische Website** bezeichnet.

Für die Aktualisierung statischer Websites sind HTML- und CSS-Kenntnisse erforderlich. Wollen die Verantwortlichen den Text auf einer Unterseite ändern, müssen Sie dazu den HTML-Code manuell aktualisieren und auf den Webserver hochladen.

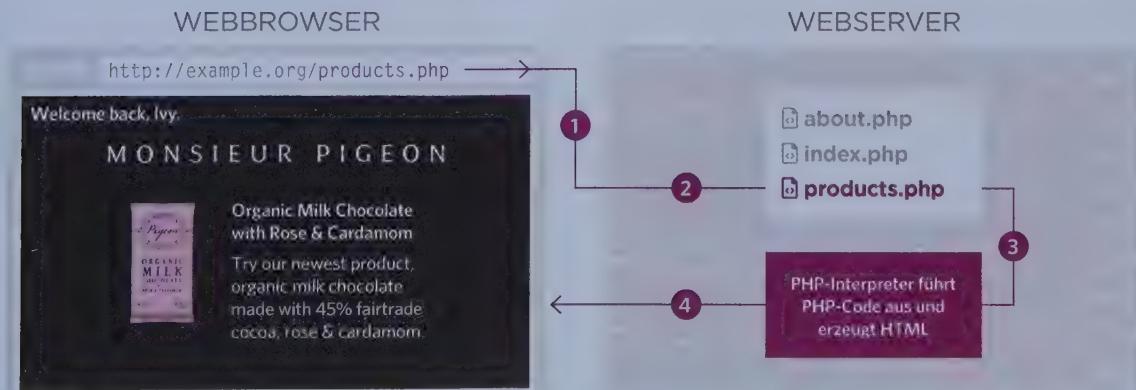


Dieses Buch setzt HTML- und CSS-Kenntnisse voraus. Wenn Ihnen diese Technologien neu sind, sollten Sie unser Buch zu diesem Thema lesen:
<http://htmlandcssbook.com>.

Werden Websites mit PHP erstellt, kann jeder Besucher einen individuellen Inhalt sehen, da die an den Browser gesendete HTML-Datei von der PHP-Seite eigens generiert wird.

Websites wie eBay, Facebook und Nachrichtenportale stellen häufig bei jedem Besuch aktualisierte Informationen bereit. Wenn Sie im Browser den Seitenquelltext betrachten, erkennen Sie zwar HTML-Code, aber es wird wohl kaum eine Programmiererin den Code zwischen Ihren Besuchen manuell aktualisiert haben. Solche Websites werden als **dynamisch** bezeichnet, weil die an den Browser zurückgesendete HTML-Seite mittels Anweisungen erzeugt wird, die in einer Sprache wie zum Beispiel PHP formuliert sind.

1. Wenn ein Browser eine Seite einer PHP-basierten Website aufruft, wird die Anfrage an den Webserver gesendet.
2. Der Webserver ruft die PHP-Datei ab.
3. Der sogenannte **PHP-Interpreter** führt sämtlichen PHP-Code innerhalb dieser Datei aus, wodurch eine HTML-Seite für diesen speziellen Besucher entsteht.
4. Der Webserver sendet die eigens für den Besucher erstellte HTML-Seite an dessen Browser. (Er speichert keine Kopie der Datei; wenn die PHP-Datei das nächste Mal aufgerufen wird, erstellt er eine neue HTML-Seite für den Besucher.)



Quelle: Microsoft Corporation, mit freundlicher Genehmigung

Der PHP-Code selbst wird nicht an den Webbrower zurückgesandt, sondern dient nur zur Erstellung einer HTML-Seite, die dann an den Browser zurückgesandt wird. Da diese Verarbeitung auf dem Webserver erfolgt, spricht man hier von **serverseitiger Programmierung**.

Mit PHP können Sie HTML-Seiten erstellen, die auf jeden einzelnen Besucher zugeschnitten oder **personalisiert** sind. Sie können beispielsweise den Namen des Besuchers oder für ihn interessante Themen oder Beiträge seiner Freunde enthalten.

PHP: DIE SPRACHE UND DER INTERPRETER

Der PHP-Interpreter ist ein Programm, das auf dem Webserver läuft. Sie können ihm mit in der Skriptsprache PHP verfasstem Code sagen, was er tun soll.

Mithilfe von Software können Menschen einen Computer für bestimmte Aufgaben einsetzen, ohne genau zu wissen, wie der Computer diese Aufgabe erledigt. Zum Beispiel:

- Mit E-Mail-Programmen können Sie E-Mails versenden und empfangen, ohne zu verstehen, wie Computer E-Mails speichern oder übertragen.
 - Mit Photoshop können Sie Bilder bearbeiten, ohne lernen zu müssen, wie Computer Bilder verändern.
- Ein Programm kann bei jedem Aufruf dieselben Aufgaben erfüllen, aber mit unterschiedlichen Daten:
- Mit einem E-Mail-Programm können E-Mails erstellt, gesendet, empfangen und archiviert werden, aber der Inhalt und der Empfänger jeder E-Mail können unterschiedlich sein.
 - Photoshop kann Filter anwenden, die Bildgröße ändern oder ein Bild zuschneiden. Diese Veränderungen können an beliebigen Bildern vorgenommen werden.

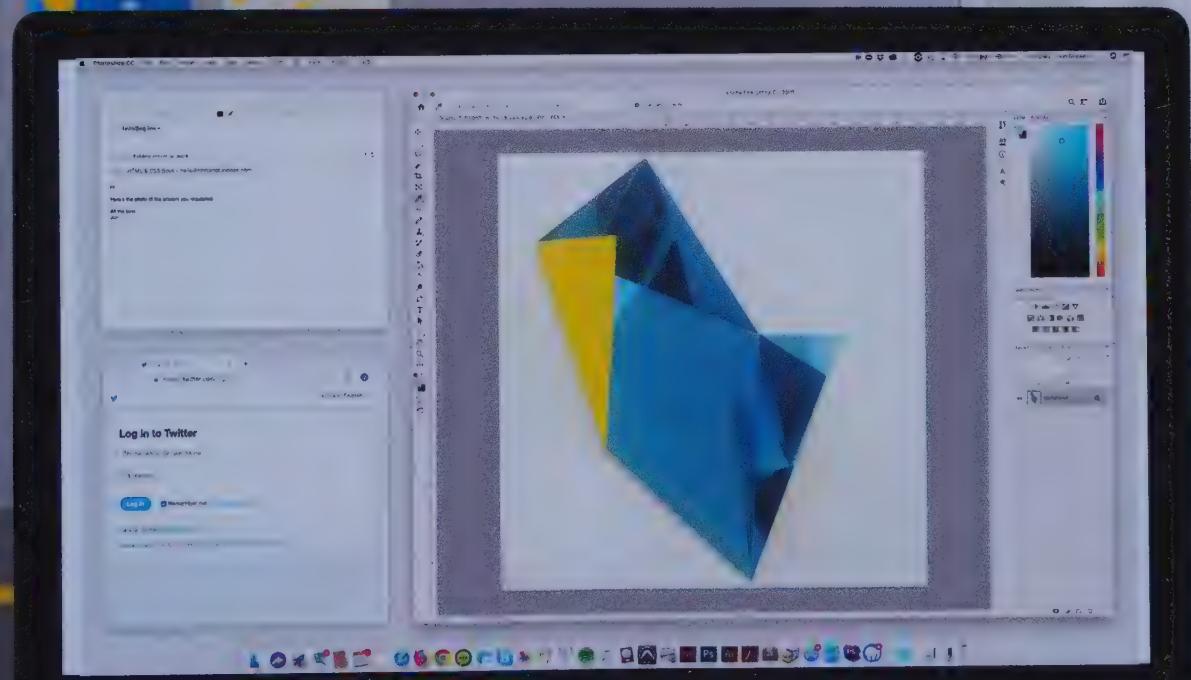
Beide Softwareprodukte verfügen über eine grafische Benutzeroberfläche, über die Sie diese Aufgaben erledigen können.

Der PHP-Interpreter ist ebenfalls ein Stück Software, das als Komponente eines Webservers läuft. Anstelle einer grafischen Benutzeroberfläche übermitteln Sie dem PHP-Interpreter Ihre Anweisungen jedoch mithilfe von Code, der in der Programmiersprache PHP geschrieben wurde.

Wenn Sie eine Webseite mit PHP erstellen, führt diese Seite immer dieselben Anweisungen aus, aber sie kann diese Anweisungen bei jeder neuen Seitenanforderung mit anderen Daten ausführen. Zum Beispiel könnte eine mit PHP erstellte Website folgende Features aufweisen:

- Eine Login-Seite, über die sich jedes Mitglied mit einer individuellen Kombination aus E-Mail-Adresse und Passwort anmeldet.
- Eine Benutzerkontoseite, auf der jedes Mitglied sämtliche Details zu seinem Konto einsehen kann. Selbst wenn Hunderte verschiedener Personen die Seite gleichzeitig nutzen, sieht jeder nur seine eigenen Daten.

Dies ist möglich, weil die Regeln oder Anweisungen zur Ausführung dieser Aufgaben für jeden Benutzer gleich sind, die Daten, die jeder Benutzer liefert oder sieht, jedoch unterschiedlich sein können.



Olafr Eliasson — Baroque

EINE AUFGABE MIT UNTERSCHIEDLICHEN DATEN AUSFÜHREN

Mit Programmiersprachen können Sie Regeln festlegen, nach denen ein Computer eine Aufgabe ausführen soll. Dabei kann das Programm bei jeder neuen Ausführung der Aufgabe andere Daten verwenden.

Bei allen Programmiersprachen müssen Sie dem Computer präzise Anweisungen geben und ihm genau sagen, was er tun soll. Diese Anweisungen unterscheiden sich stark von solchen, die Sie einer Person geben würden, wenn Sie sie bitten, eine Aufgabe auszuführen. Stellen Sie sich vor, Sie möchten fünf Schokoriegel kaufen und müssen den Gesamtpreis der Riegel ausrechnen. Um den Gesamtbetrag zu ermitteln, multiplizieren Sie den Preis eines einzelnen Riegels mit der Anzahl der Riegel, die Sie kaufen möchten. Diese Regel könnten Sie folgendermaßen ausdrücken:

$$\text{Gesamtbetrag} = \text{Einzelpreis} \times \text{Menge}$$

Und so berechnen sich die Kosten für die Süßigkeiten:

- Wenn ein Schokoriegel 1\$ kostet und Sie 5 Stück davon kaufen, liegt der Gesamtbetrag bei 5\$.
- Wenn ein einzelner Riegel 1,50\$ kostet, gilt die gleiche Regel, aber der Gesamtbetrag beliefe sich auf 7,50\$.
- Wenn Sie 10 Riegel zu je 2\$ kaufen möchten, gilt die gleiche Regel, aber der Gesamtpreis würde 20\$ betragen.

Die Werte, die anstelle der Wörter Gesamtbetrag, Einzelpreis und Menge eingesetzt werden, können sich ändern, aber die Regel zur Berechnung des Gesamtpreises der Süßwaren bleibt gleich.

Wenn Sie eine Webseite mit PHP erstellen, müssen Sie zunächst Folgendes herausfinden:

- welche Aufgaben Sie ausführen möchten
- welche Daten sich bei jeder Ausführung der Aufgabe verändern könnten

Dann geben Sie dem PHP-Interpreter detaillierte Anweisungen, wie er die Aufgabe durchführen soll, und verwenden Namen, um die veränderlichen Werte darzustellen.

Angenommen, Sie sagen dem PHP-Interpreter:

$$\text{Einzelpreis} = 3$$

$$\text{Menge} = 5$$

Und verwenden dabei diese Regel:

$$\text{Gesamtbetrag} = \text{Einzelpreis} \times \text{Menge}$$

Dann entspräche das Wort Gesamtbetrag einem Wert von 15.

Beim nächsten Seitenaufruf können Sie einen anderen Einzelpreis oder eine andere Menge eingeben, und der neue Gesamtbetrag wird nach der gleichen Regel berechnet.

Wörter, die Werte repräsentieren, werden in der Programmierung als **Variablen** bezeichnet, weil sich der von ihnen dargestellte Wert bei jeder Programm-ausführung verändern (oder variieren) kann.

Gesamtbetrag =
Einzelpreis x Menge



\$9 = 3 x 3

WAS IST EINE PHP-SEITE?

Eine PHP-Seite enthält häufig eine Mischung aus HTML- und PHP-Code. Sie kann dazu dienen, eine HTML-Seite zurück an den Browser zu senden.

Unten links sehen Sie eine PHP-Seite mit einem Mix aus HTML- und PHP-Code.

- HTML-Code ist blau dargestellt
- PHP-Code ist violett dargestellt

Das geschieht, wenn der PHP-Interpreter die Datei öffnet:

- Er kopiert sämtlichen HTML-Code direkt in eine temporäre HTML-Datei, die er für den Besucher anlegt.
- Er befolgt sämtliche PHP-Anweisungen (diese erzeugen häufig die Inhalte der HTML-Seite).

Der hier dargestellte PHP-Code ermittelt das aktuelle Jahr und gibt es innerhalb der öffnenden `<p>`- und schließenden `</p>`-Tags aus.

PHP-Code kann einfache Aufgaben wie Rechenoperationen oder die Abfrage des aktuellen Datums erfüllen, aber auch komplexere Aufgaben, wie etwa die über ein HTML-Formular übermittelten Informationen zur Aktualisierung der in einer Datenbank hinterlegten Daten zu nutzen.

Sobald der PHP-Interpreter die Verarbeitung der PHP-Datei abgeschlossen hat, sendet er die für den Besucher erstellte temporäre HTML-Seite zurück an den Browser und verwirft sie anschließend.

Unten rechts sehen Sie die HTML-Seite, die an den Browser zurückgeschickt wird, nachdem der PHP-Interpreter den PHP-Code abgearbeitet hat.

Der PHP-Interpreter hat das aktuelle Jahr ermittelt und es in der erzeugten HTML-Seite ausgegeben.

```
PHP
<!DOCTYPE html>
<html>
  <body>
    <h1>Current Year</h1>
    <p>It is: <?php echo date('Y'); ?></p>
  </body>
</html>
```

```
HTML
<!DOCTYPE html>
<html>
  <body>
    <h1>Current Year</h1>
    <p>It is 2021</p>
  </body>
</html>
```

Der PHP-Interpreter ermittelt das aktuelle Jahr und schreibt es im Klartext zwischen die Absatz-Tags.

Eine Seite erfüllt meist bei jedem neuen Aufruf dieselbe Aufgabe, kann dabei aber für verschiedene Seitenbesucher auf unterschiedliche Informationen zurückgreifen.

Eine PHP-Website besteht aus mehreren PHP-Seiten die jeweils eine spezifische Aufgabe erfüllen. Eine Website mit Login-Möglichkeit für Mitglieder könnte beispielsweise über folgende Unterseiten verfügen:

- Login-Seite – über diese können sich Mitglieder auf der Website anmelden
 - Profil-Seite – um die Mitgliederprofile anzuzeigen
- Diese Seiten müssen bei jedem Aufruf in der Lage sein, auf andere Daten zurückzugreifen, die für das aktuelle Mitglied relevant sind. Daher muss die Seite:
- Anweisungen zur Durchführung der von der Seite zu erfüllenden Aufgabe enthalten.
 - jedem Datenelement, das sich beim Seitenaufruf verändern kann, einen eigenen Namen zuordnen.

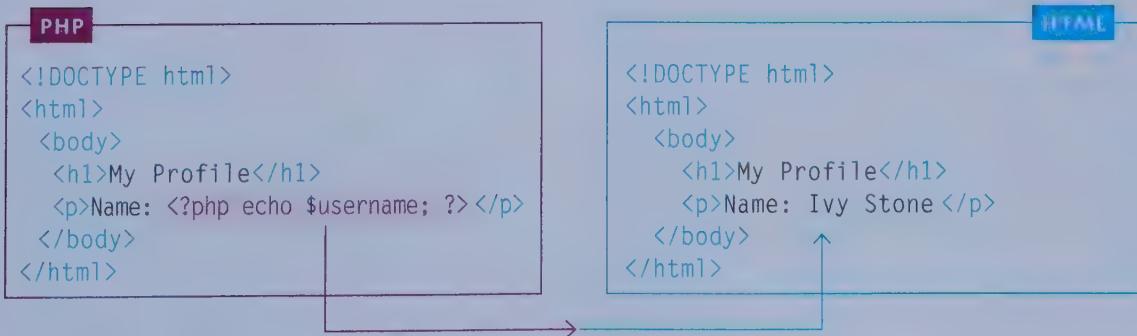
In PHP nutzen Variablen einen Namen, um einen Wert darzustellen, der sich bei jedem neuen Seitenaufruf verändern kann.

Der PHP-Code kann den PHP-Interpreter anweisen:

- welcher Variablenname für ein Datenelement verwendet werden soll, das sich bei jedem Seitenaufruf ändern kann.
- welcher Wert bei diesem speziellen Seitenaufruf verwendet werden soll.

Nachdem die HTML-Seite an den Benutzer zurückgeschickt wurde, vergisst der PHP-Interpreter alle in den Variablen gespeicherten Werte, sodass er die gleiche Aufgabe für die nächste Person, die die Seite anfordert, durchführen kann (dann aber mit anderen Werten).

Um Daten länger zu speichern, legen Sie sie in einer Datenbank wie MySQL ab, die Sie auf der nächsten Seite kennenlernen.



Der PHP-Interpreter ruft den in der Variablen \$username gespeicherten Wert ab und gibt ihn innerhalb der Absatz-Tags aus.

WAS IST MYSQL?

MySQL ist eine bestimmte Art von Datenbank. Datenbanken speichern Daten strukturiert ab, sodass die darin enthaltenen Informationen leicht zugänglich sind und aktualisiert werden können.

Tabellenkalkulationsprogramme wie Excel speichern Informationen in einem Raster aus Spalten und Zeilen. Mit den in der Tabelle gespeicherten Daten können sie zudem auch Berechnungen durchführen oder sie mithilfe von Formeln manipulieren.

MySQL ist eine Software, die Informationen auf ähnliche Weise speichert: in **Tabellen**, die ebenfalls aus Spalten und Zeilen bestehen. Mittels PHP können Sie auf die in der Datenbank gespeicherten Informationen zugreifen und diese aktualisieren.

Eine Datenbank kann aus mehreren Tabellen bestehen. Jede Tabelle enthält in der Regel einen Datentyp, den die Website speichern muss. Im Folgenden finden Sie zwei Beispiele für Datenbanktabellen mit diesen Inhalten:

- Mitglieder (oder Benutzer) einer Website
- Artikel, die eine Website anzeigen

Innerhalb jeder Tabelle beschreiben die **Spaltennamen** die Art der in der jeweiligen Tabellenspalte enthaltenen Informationen:

- Die member-Tabelle speichert den Vor- und Nachnamen, die E-Mail-Adresse, das Passwort, das Beitrittsdatum und das Profilbild eines jeden Mitglieds.
- Die article-Tabelle speichert den Titel, eine Zusammenfassung, den Inhalt, das Erstellungsdatum eines jeden Artikels und einige zusätzliche Werte, die auf der rechten Seite erläutert werden.

Die einzelnen **Zeilen** enthalten die Daten zur Beschreibung eines der Tabelleneinträge:

- In der member-Tabelle steht jede Zeile für ein Mitglied.
- In der article-Tabelle enthält jede Zeile einen Artikel.

TABELLENNAME		SPALTENNAME		SPALTE	
member					
id	forename	surname	email	password	joined
1	Ivy	Stone	ivy@eg.link	\$2y\$10\$MAdTTCAOMi0w	2021-01-01 20:28:47
2	Luke	Wood	luke@eg.link	\$2y\$10\$NN5HEAD3atar	2021-01-02 09:17:21
3	Emiko	Ito	emi@eg.link	\$2y\$10\$/RpRmIUMStji	2021-01-02 10:42:36
					picture
					ivy.jpg
					NULL
					emi.jpg

article									
id	title	summary	content	created	category_id	member_id	image_id	published	
1	Systemic Brochure	<p>This	<p>The brochure is a comprehensive guide to our company's products and services. It highlights our latest innovations and how they can benefit your business.	2021-01-01	1	2	1	1	
2	Polite Poster	<p>These	<p>The poster is a simple yet effective way to promote our products. It features large, bold text and a clear call to action.	2021-01-02	1	1	2	1	—
3	Swimming Architect	<p>This	<p>The architect is a detailed plan for a new swimming pool. It includes blueprints, materials lists, and construction timelines.	2021-01-02	4	1	3	1	

ZEILE

Mithilfe von PHP können Sie:

- **Daten aus der Datenbank abrufen** und diese Informationen auf einer Webseite anzeigen.
- **Neue Zeilen mit Daten hinzufügen.** Um einen neuen Artikel zu erstellen, fügen Sie der Artikeltabelle eine Zeile hinzu und geben an, welche Daten in den einzelnen Spalten gespeichert werden sollen.
- **Datenzeilen löschen.** Um einen Artikel zu löschen, entfernen Sie die gesamte Zeile, die den Artikel darstellt.
- **Daten in einer vorhandenen Zeile ändern.** Um die E-Mail-Adresse eines Mitglieds zu aktualisieren, suchen Sie in der Mitgliedertabelle die Zeile, die das Mitglied repräsentiert, und aktualisieren dann den Wert in der email-Spalte dieser Zeile.

Jede Tabellenzeile besitzt in dieser Spalte einen eindeutigen Wert (hier beginnen die Werte bei 1 und erhöhen sich in jeder Zeile um 1). Mit dem id-Wert können Sie der Datenbank mitteilen, mit welcher Dateneinheit Sie arbeiten wollen. Sie könnten zum Beispiel das Mitglied mit der ID 2 oder den Artikel mit der ID 1 abrufen.

MySQL ist eine sogenannte **relationale Datenbank**, weil sie die Beziehungen zwischen den in den verschiedenen Tabellen gespeicherten Datentypen herstellen kann.

In den nachstehenden Tabellen wurden die Artikel beispielweise von verschiedenen Mitgliedern der Webseite verfasst. In der article-Tabelle gibt der Wert in der Spalte member_id Aufschluss darüber, welcher Benutzer den Artikel geschrieben hat, da er eine Zahl enthält, die mit einem der Werte in der Spalte id der member-Tabelle übereinstimmt.

Der erste Artikel wurde von dem Mitglied verfasst, dessen id-Spalte den Wert 2 enthält (Luke Wood). Der zweite und der dritte Artikel wurden von der Benutzerin verfasst, deren id-Spalte den Wert 1 hat (Ivy Stone).

Für diese Relationen gilt:

- Sie strukturieren die Daten und stellen damit sicher, dass jede Tabelle nur eine bestimmte Art von Daten enthält (member oder article).
- Sie verhindern, dass die Datenbank dieselben Daten in mehreren Tabellen speichern muss (was Platz in der Datenbank spart).
- Sie machen es einfacher, den Datenstand aktuell zu halten. Wenn ein Mitglied seinen Namen ändert, muss dieser nur in der Mitgliedertabelle aktualisiert werden (und nicht in jedem von ihm verfassten Artikel).

member								
id	forename	surname	email	password	joined	picture		
1	Ivy	Stone	ivy@eg.link	\$2y\$10\$MAdTTCA0Mi0w	2021-01-01 20:28:47	ivy.jpg		
2	Luke	Wood	luke@eg.link	\$2y\$10\$NN5HEAD3atar	2021-01-02 09:17:21	NULL		
3	Emiko	Ito	emi@eg.link	\$2y\$10\$/RpRmiUMStji	2021-01-02 10:42:36	emi.jpg		

article								
id	title	summary	content	created	category_id	member_id	image_id	published
1	Systemic Brochure	<p>This	2021-01-01	1	2	1	1	
2	Polite Poster	<p>These	2021-01-02	1	2	2	1	
3	Swimming Architect	<p>This	2021-01-02	4	1	3	1	

DIE GESCHICHTE VON PHP

Wie bei der meisten Software gibt es auch von PHP und MySQL zahlreiche Versionen. Neuere Versionen haben mehr Funktionen und laufen schneller als ältere Versionen.

PHP wurde 1994 von Rasmus Lerdorf entwickelt. Er gab den Code 1995 öffentlich frei und ermutigte die Benutzer, ihn zu verbessern. Damals standen die Buchstaben noch für Personal Home Page. Heute steht das Akronym für PHP: Hypertext Processor.

PHP kommt heute auf 80 % aller Websites zum Einsatz, die eine serverseitige Programmiersprache verwenden.

Websites wie Facebook, Etsy, Flickr und Wikipedia wurden alle ursprünglich in PHP entwickelt (auch wenn einige inzwischen auch andere Technologien einsetzen).

Beliebte Open-Source-Projekte wie WordPress (das bei über 35 % aller Websites weltweit im Hintergrund werkelt), Drupal, Joomla und Magento sind allesamt in PHP geschrieben. Wenn Sie PHP lernen, können Sie besser damit arbeiten.

In jeder neuen PHP-Version kommen zusätzliche Funktionen hinzu. Dieses Buch beschreibt Funktionen bis einschließlich PHP 8, das im November 2020 veröffentlicht wurde.

PHP 1	1995
PHP 2	1996
PHP 3	1997
PHP 4	1998
PHP 5	1999
PHP 5.1	2000
PHP 5.2	2001
PHP 5.3	2002
PHP 5.4	2003
PHP 5.5	2004
PHP 5.6	2005
PHP 7	2006
PHP 7.1	2007
PHP 7.2	2008
PHP 7.3	2009
PHP 7.4	2010
PHP 8	2011
PHP 8.0	2012
PHP 8.1	2013
PHP 8.2	2014
PHP 8.3	2015
PHP 8.4	2016
PHP 8.5	2017
PHP 8.6	2018
PHP 8.7	2019
PHP 8.8	2020
PHP 8.9	2021

DIE GESCHICHTE VON MYSQL

1995	MYSQL 1
1996	
1997	MYSQL 3.2
1998	PHPMYADMIN
1999	
2000	
2001	
2002	
2003	MYSQL 4
2004	
2005	
2006	MYSQL 5
2007	
2008	SUN KAUFT MYSQL
2009	MYSQL 5.1
2010	MARIADB
2010	ORACLE KAUFT SUN
2011	MYSQL 5.5
2012	
2013	MYSQL 5.6
2014	
2015	
2016	MYSQL 5.7
2017	
2018	MYSQL 8
2019	
2020	
2021	

MySQL wurde erstmals 1995 veröffentlicht. Die Buchstaben SQL (ausgesprochen entweder ess-qu-ell oder siquell) stehen für Structured Query Language. SQL ist eine Sprache, die verwendet wird, um Informationen in relationalen Datenbanken hinein zu bekommen und sie wieder daraus abzurufen.

MySQL wurde von dem schwedischen Unternehmen MySQL AB entwickelt und letztlich kostenfrei zur Verfügung gestellt. Michael Widenius, einer der Schöpfer von MySQL, benannte die Software nach seiner Tochter, My.

MySQL AB wurde im Januar 2008 an Sun Microsystems verkauft, das wiederum 2010 von Oracle übernommen wurde.

Als die MySQL-Entwickler erfuhren, dass Oracle Sun (und damit auch MySQL) kaufen würde, waren sie besorgt, dass die Datenbank möglicherweise nicht mehr frei verfügbar bleiben würde, und entwickelten daher die Open-Source-Version MariaDB (benannt nach der jüngeren Tochter des Gründers, Maria).

Websites wie Facebook, YouTube, Twitter, Netflix, Spotify und Wordpress verwenden alle entweder MySQL oder MariaDB.

phpMyAdmin ist ein Tool zur Verwaltung von MySQL- und MariaDB-Datenbanken. Es wurde 1998 als kostenloses Werkzeug zur Verwaltung von MySQL-Datenbanken veröffentlicht (und es funktioniert auch mit MariaDB).

Der Code in diesem Buch funktioniert mit MySQL ab Version 5.5 oder MariaDB ab Version 5.5 und wird über phpMyAdmin ausgeführt.

Die neueste Version von MySQL ist zum Zeitpunkt der Erstellung dieses Buchs Version 8. (MySQL 6 wurde nie veröffentlicht, und Version 7 ist nicht aufgeführt, da sie für den Einsatz auf Server-Clustern und nicht, auf Ihrem PC konzipiert wurde.)

DAS FINDEN SIE IN DIESEM BUCH

Das vorliegende Buch ist in vier Abschnitte unterteilt. Hier ist eine Übersicht der Themen, die Sie in den einzelnen Abschnitten lernen.

A: GRUNDLAGEN DER PROGRAMMIERUNG

Im ersten Abschnitt erfahren Sie, wie Sie mittels PHP-Code Anweisungen schreiben, denen der PHP-Interpreter folgen kann. Sie lernen dort:

- grundlegende Programmieranweisungen
- in unterschiedlichen Situationen unterschiedlichen Code ausführen (z. B. einen bestimmten Code ausführen, wenn ein Benutzer angemeldet ist, und einen anderen Code, wenn er nicht angemeldet ist)
- wie Funktionen den gesamten Code zusammenfassen können, der zur Ausführung einer bestimmten Aufgabe erforderlich ist
- wie Sie mittels Klassen und Objekten den Code so strukturieren, dass er Dinge aus der Welt um uns herum darstellen kann

B: DYNAMISCHE WEBSEITEN

Der zweite Buchteil behandelt eine Reihe von PHP-Werkzeugen, mit denen Sie dynamische Webseiten erstellen können. Sie lernen dabei, wie man:

- von einem Webbrowser gesendete Daten abruft
- überprüft, ob die Benutzer die erforderlichen Daten im korrekten Format übermittelt haben
- die gesendeten Daten weiterverarbeitet
- vom Benutzer hochgeladene Dateien weiterverarbeitet
- Datums- und Zeitangaben in PHP darstellt
- Daten vorübergehend in Cookies und Sessions speichert
- Probleme mit dem Code erkennt und behebt

C: DATENBANKGESTÜTZTE WEBSITES

Der dritte Abschnitt zeigt auf, wie Sie Daten aus einer Datenbank abrufen und auf Ihren Webseiten anzeigen und wie Sie die in der Datenbank gespeicherten Daten aktualisieren. Sie lernen dabei, wie:

- Datenbanken ihre Daten speichern
- die Sprache SQL eingesetzt wird, um die Daten in einer Datenbank abzurufen oder zu aktualisieren
- Daten aus der Datenbank auf eine PHP-Seite angezeigt werden
- Besuchern über HTML-Formulare ermöglicht wird, die in der Datenbank gespeicherten Daten zu aktualisieren

D: DIE BEISPIELANWENDUNG AUSBAUEN

Der vierte Abschnitt beschreibt praktische Techniken zur Erstellung von Websites und Webanwendungen in PHP. Die Beispielanwendung ist ein einfaches Content-Management-System mit Social-Media-Funktionen. Sie lernen, wie Sie:

- Ihren Code besser strukturieren
- Code von anderen Programmierern einbinden
- E-Mails mittels PHP versenden
- Mitgliedern die Registrierung und Anmeldung auf einer Website ermöglichen
- auf die einzelnen Mitglieder zugeschnittene Seiten erstellen
- suchmaschinenfreundliche URLs einsetzen
- soziale Funktionen wie Likes und Kommentare einbauen



GRUNDLAGEN DER PROGRAM- MIERUNG

Im ersten Buchteil lernen Sie die Grundlagen, die Sie zum Schreiben von PHP-Code benötigen.

Beim Programmieren erstellen Sie eine Reihe von Anweisungen, die ein Computer befolgen kann, um eine bestimmte Aufgabe zu erfüllen. Diese Anweisungen sind vergleichbar mit einem Kochrezept, das alle Schritte für die Zubereitung eines Gerichts enthält. Eine einzelne Anweisung wird in PHP auch als **Statement** bezeichnet.

Da PHP entwickelt wurde, um dynamische Websites zu ermöglichen, die für jeden Besucher neue HTML-Seiten erstellen, konzentrieren sich die in diesem ersten Buchteil behandelten Anweisungen auf die Erstellung solcher HTML-Seiten mit PHP.

Eine komplette Website umfasst oft mehrere Tausend Zeilen Code, daher ist es besonders wichtig, Ihren Code sorgfältig zu strukturieren. In diesem Abschnitt werden zwei Konzepte vorgestellt, um zusammengehörige Anweisungen zu gruppieren:

- **Funktionen** fassen die Statements zusammen, die zur Ausführung einer bestimmten Aufgabe erforderlich sind.
- **Objekte** fassen eine Reihe von Statements zusammen, die gemeinsam bestimmte Konzepte repräsentieren, wie etwa Artikel, die auf einer Website angezeigt werden, Produkte, die eine Website verkauft, oder Mitglieder, die sich auf einer Website registriert haben.

Die Themen in diesem Abschnitt liefern die Grundlagen für alles Weitere, das Sie in diesem Buch lernen.

Ehe Sie sich in das erste Kapitel vertiefen, sollten Sie sich jedoch einige hilfreiche Grundkenntnisse aneignen.

SOFTWARE UND CODE-BEISPIELE INSTALLIEREN

Um auf Ihrem Laptop oder Desktop-PC Websites mit PHP und einer Datenbank wie MySQL zu erstellen, müssen Sie einige Anwendungen installieren. Sobald die erforderlichen Programme installiert sind, müssen Sie die Code-Beispiele für dieses Buch von unserer Website herunterladen: <http://phpandmysql.com/code/>

PHP-DATEIEN BESTEHEN AUS EINER MISCHUNG AUS HTML-UND PHP-CODE

Da PHP zur Erstellung von HTML-Seiten verwendet wird, beinhaltet eine PHP-Seite oft einen Mix aus HTML- und PHP-Code. Deshalb müssen Sie lernen, wie der PHP-Interpreter zwischen diesen beiden Arten von Code unterscheiden kann.

PHP DIENT ZUR ERSTELLUNG VON HTML

Eine Ihrer häufigsten Anweisungen an den PHP-Interpreter ist es, der HTML-Seite, die an den Besucher zurückschickt wird, Inhalte hinzuzufügen. Diese Anweisung nutzen wir in jedem Beispiel in diesem Buchteil.

DEN PHP-CODE KOMMENTIEREN

Kommentare werden vom PHP-Interpreter selbst nicht ausgeführt, aber sie helfen Ihnen (und anderen) zu verstehen, was der Code bezeichnen soll. Daher ist es wichtig zu lernen, wie man sie einfügt. Erklärende Kommentare finden sich in diesem Buch in allen Code-Beispielen.

SOFTWARE UND DATEIEN INSTALLIEREN

Die unten aufgeführten Tools installieren die gesamte Software, die Sie zur Erstellung datenbankgestützter Websites auf Ihrem Desktop- oder Laptop-Computer benötigen.

Zur Arbeit mit diesem Buch müssen Sie Folgendes installieren:

- Einen **Webserver**, auf dem ein PHP-Interpreter läuft; in diesem Buch verwenden wir Apache (der am weitesten verbreitet ist)
- die Datenbank-Software **MySQL** oder **MariaDB**
- **phpMyAdmin** zur Verwaltung der Datenbank

Statt jedes dieser Programme einzeln herunterzuladen und zu installieren, können Sie mit den unten beschriebenen Tools alle Programme auf einmal herunterladen und installieren.

Zudem empfehlen wir die Verwendung eines Code-Editors, wie er hier beschrieben ist:

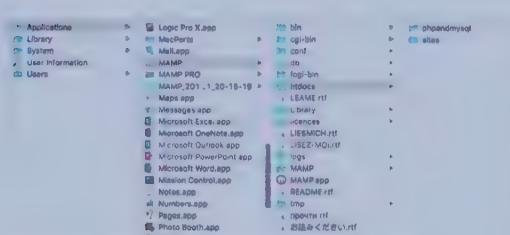
<http://notes.re/php/editors>

INSTALLATION AUF EINEM MAC

Mac-Nutzern empfehlen wir, die erforderliche Software mit dem Tool MAMP zu installieren. Den Download-Link und eine Anleitung finden Sie hier: <http://notes.re/php/mamp>

Bei der Installation von MAMP auf einem Mac (mit den Standardeinstellungen) wird der Ordner /Programme/MAMP angelegt.

Darin befindet sich ein weiterer Ordner htdocs. All Ihre PHP-Webseiten müssen in diesem Ordner abgelegt werden. Dieser Ordner wird auch **Stammverzeichnis (document root)** genannt.



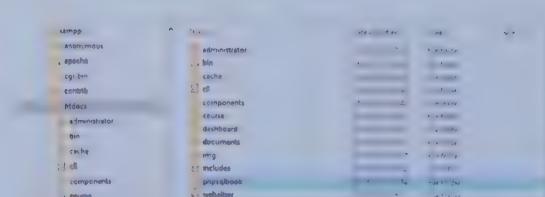
Quelle: Microsoft Corporation, mit freundlicher Genehmigung

INSTALLATION AUF WINDOWS-ODER LINUX-PC

Windows- und Linux-Nutzern empfehlen wir die Installation der erforderlichen Software über das Tool XAMPP. Den Download-Link und eine Anleitung finden Sie hier: <http://notes.re/php/xamp>.

Bei der Installation von XAMPP auf einem Windows-PC (mit den Standardeinstellungen) wird der Ordner c:\xampp\ angelegt.

Darin befindet sich ein weiterer Ordner htdocs. All Ihre PHP-Webseiten müssen in diesem Ordner abgelegt werden. Dieser Ordner wird auch Stammverzeichnis (document root) genannt.



Quelle: Microsoft Corporation, mit freundlicher Genehmigung

DIE CODE-BEISPIELE HERUNTERLADEN

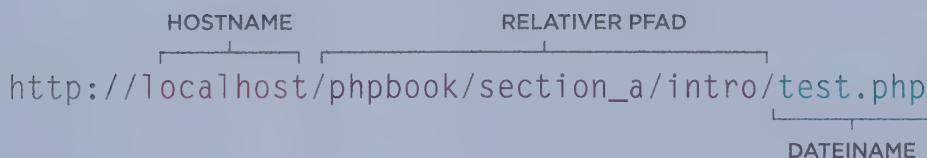
Laden Sie sich die Code-Beispiele für das Buch mit diesem Download-Link herunter: <http://phpandmysql.com/code>.

Der Beispielcode liegt in dem Ordner `phpbook`. Dieser enthält Unterordner für jeden Buchabschnitt sowie für jedes einzelne Kapitel. Legen Sie den Ordner `phpbook` im Ordner `htdocs` ab.

Um PHP-Dateien in Ihrem Browser zu öffnen, müssen Sie eine URL in die Adressleiste eingeben (wenn Sie eine Datei über den **Öffnen**-Befehl **Datei**-Menü, durch einen Doppelklick oder durch Herüberziehen in den Browser öffnen, wird der PHP-Code nicht ausgeführt).

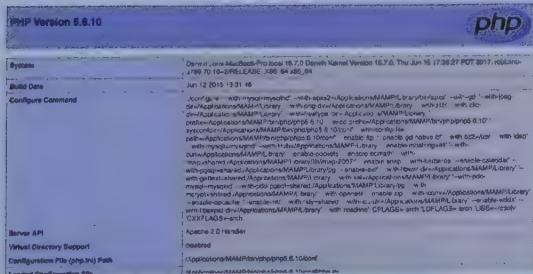
Wenn Sie versuchen, die unten stehende URL zu öffnen, sollte Ihnen die darunter angezeigte Testseite dargestellt werden. Geben Sie beim Öffnen von Dateien auf Ihrem Computer **localhost** statt eines Domänen-namens ein. Danach folgt der Pfad vom Ordner `htdocs` zu der Datei, die Sie öffnen möchten.

Der unten angegebene Pfad veranlasst den Server, im Ordner `phpbook/section_a/intro` nach der Datei `test.php` zu suchen.



Welcome to PHP & MySQL

Your web server is working



PROBLEMBEHANDLUNG

Wenn Sie die links dargestellte Seite nicht sehen können, lesen Sie hier nach:

<http://notes.re/php/mamp> für MAMP

<http://notes.re/php/xampp> für XAMPP

MAMP benötigt manchmal eine **Portnummer**. Es verwendet Port 8888, also geben Sie Folgendes ein:

`http://localhost:8888`

Mithilfe von Portnummern können mehrere auf einem Computer installierte Programme dieselbe Internet-verbindung nutzen. Das ist ähnlich wie in Unternehmen mit einer bestimmten Telefonnummer, gefolgt von den Durchwahlnummern der einzelnen Arbeitsplätze.

AUF PHP-SEITEN HTML- UND PHP-CODE MISCHEN

Viele PHP-Seiten mischen HTML- und PHP-Code. Der PHP-Code steht zwischen PHP-Tags. Die öffnenden und schließenden PHP-Tags sowie der darin enthaltene PHP-Code werden als PHP-Block bezeichnet.

OFFNENDES TAG



Das öffnende Tag zeigt an, dass der PHP-Interpreter mit der Verarbeitung des Codes beginnen muss, bevor irgendwelche Inhalte an den Browser zurückgesendet werden.

SCHLIESSENDES TAG



Das schließende Tag zeigt an, dass der PHP-Interpreter die Verarbeitung des Codes beenden kann, bis er auf ein weiteres öffnendes <?php-Tag trifft.

PHP ist eine Art von Programmiersprache, die als Skriptsprache bezeichnet wird. Skriptsprachen sind so konzipiert, dass sie in einer bestimmten Umgebung laufen; PHP wurde für die Ausführung durch einen PHP-Interpreter auf einem Webserver entwickelt. Eine einzelne PHP-Seite wird oft als Skript bezeichnet.

Sie sollten bei PHP-Code die Groß- und Kleinschreibung beachten.

Es gibt zwar einige Teile der Sprache, die nicht zwischen Groß- und Kleinschreibung unterscheiden, aber wenn Sie so tun, als würden sie alle zwischen Groß- und Kleinschreibung unterscheiden, verringern Sie die Fehlergefahr.

Eine PHP-Seite ist eine Textdatei (genau wie eine HTML-Datei). Die zugehörige Dateiendenerweiterung lautet .php. Sie weist den Webserver an, die Datei an den PHP-Interpreter zu senden, damit dieser die in PHP geschriebenen Anweisungen abarbeiten kann.

Unten sehen Sie eine PHP-Seite mit folgendem Inhalt:

- **PHP-Code zwischen den violett gefärbten PHP-Tags**. Der PHP-Interpreter verarbeitet jeglichen Code, der sich innerhalb der PHP-Tags befindet.
- **HTML-Code außerhalb der weiß dargestellten PHP-Tags**. Dieser wird automatisch an die HTML-Datei angehängt, die an den Browser gesendet wird (der PHP-Interpreter braucht diesen Code nämlich nicht zu verarbeiten).

```
<?php  
$price = 3;  
$quantity = 5;  
$total = $price * $quantity;  
<?>  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Cost of Candy</title>  
  </head>  
  <body>  
    <p>Total: <?php echo $total; ?></p>  
  </body>  
</html>
```

Jede einzelne Anweisung innerhalb der PHP-Tags wird als Statement bezeichnet. Die meisten Statements beginnen auf einer neuen Zeile und enden mit einem Semikolon. In folgenden Fällen können Sie das Semikolon nach einer Anweisung weglassen:

- in der letzten Zeile eines PHP-Blocks
- wenn der PHP-Block nur ein Statement enthält

Indem Sie nach jedem Statement ein Semikolon setzen, können Sie Fehler vermeiden.

Diese Seite berechnet die Kosten für 5 Beutel Süßigkeiten zu je \$3 und speichert den Preis in der Variablen \$total. Anschließend wird dieser Wert in der HTML-Seite ausgegeben. Wie der PHP-Code das macht, erfahren Sie in Kapitel 1.

OFFNENDES PHP-TAG

STATEMENT

STATEMENT

STATEMENT

SCHLIESSENDES PHP-TAG

PHP-BLOCK

PHP-BLOCK

WIE PHP TEXT UND HTML AN DEN BROWSER SENDET

Der Befehl echo weist den PHP-Interpreter an, der für den Browser generierten HTML-Seite Text oder Markup hinzuzufügen.

Sämtlicher Text und/oder HTML-Code, der in Anführungszeichen nach einem echo-Befehl steht, wird an den Browser gesendet, damit er auf der Seite angezeigt werden kann. Sie können hier einfache oder doppelte Anführungszeichen verwenden, aber das öffnende und das schließende Anführungszeichen müssen übereinstimmen.

Das erste Anführungszeichen teilt dem PHP-Interpreter mit, wo der Text anfängt, den er der Seite hinzufügen soll; das zweite sagt ihm, wo er wieder endet. Der Text wird als **String**- oder **Zeichenkettenliteral** bezeichnet. Das Semikolon am Zeilenende signalisiert dem PHP-Interpreter das Ende des Statements.

```
echo '<b>Hello!</b>'
```

IN DIE BROWSERSEITE
SCHREIBEN

ANZUZEIGENDER TEXT & MARKUP

Um ein Anführungszeichen im an den Browser übermittelten Text anzuzeigen, fügen Sie einen Backslash direkt davor ein. Dieses Maskierungszeichen weist den PHP-Interpreter an, das nachfolgende Anführungszeichen nicht als Code zu behandeln. Programmierer sprechen hier auch vom **Maskieren** des Anführungszeichens.

Im Folgenden nutzt der echo-Befehl doppelte Anführungszeichen, um einen HTML-Link auszuschreiben. Die URL im href-Attribut muss in Anführungszeichen stehen, daher werden diese Anführungszeichen maskiert. Der unten dargestellte Code gibt folgenden HTML-Link aus:

```
echo "<a href =\"http://notes.re/php\">>PHP</a>";
```

ÖFFNENDES ANFÜHRUNGSZEICHEN FÜR Echo-BEFEHL

MASKIERTE ANFÜHRUNGSZEICHEN UMSCHLIESSEN DAS ATTRIBUT

SCHLIESSENDES ANFÜHRUNGS-
ZEICHEN FÜR Echo-BEFEHL

Wenn Sie den Text und das HTML für die Ausgabe in einfache Anführungszeichen setzen, können Sie auch doppelte Anführungszeichen ausgeben, ohne diese zu maskieren.

Das funktioniert, weil der PHP-Interpreter ein passendes einfaches Anführungszeichen für das Textende erwartet.

```
echo '<a href="http://notes.re/php">PHP</a>';
```

ÖFFNENDES ANFÜHRUNGSZEICHEN FÜR Echo-BEFEHL

HTML-ATTRIBUT IN
DOPPELTEM ANFÜHRUNGS-
ZEICHEN

SCHLIESSENDES AN-
FÜHRUNGSZEICHEN
FÜR Echo-BFFFHI

INHALTE IN DIE SEITE SCHREIBEN

PHP

```
<!DOCTYPE html>
<html>
  <head>
    <title>echo Command</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2><?php echo 'Ivy\'s'; ?> page</h2>
    <?php echo '<p class="offer">Offer: 20% off</p>' ?>
  </body>
</html>
```

①
②

ERGEBNIS



Hinweis: Wenn Sie nach einem echo-Befehl doppelte Anführungszeichen verwenden, prüft der PHP-Interpreter, ob der Text Variablennamen enthält (die Sie auf den Seiten 32 bis 36 kennenlernen). Wenn ja, gibt er den aktuellen Wert der Variablen aus. Bei einfachen Anführungszeichen geschieht dies nicht. Ein Beispiel hierfür finden Sie auf Seite 52.

Für den Code-Kasten auf der linken Seite gilt:

- Der Dateipfad rechts oben entspricht der Datei im Code-Download.
- Die Zahlen entsprechen den Schritten der nachfolgenden Code-Beschreibung.
 1. Der echo-Befehl verwendet einfache Anführungszeichen, um den Namen des Besuchers, gefolgt von 's', zu schreiben. Mit dem Backslash-Zeichen wird das ' zwischen dem Namen und dem Buchstaben s maskiert.
 2. Der echo-Befehl fügt einen Absatz in die Seite ein. Das <p>-Element hat ein Class-Attribut.

Da der in die Seite geschriebene Text und das Markup in einfachen Anführungszeichen stehen, stellen die doppelten Anführungszeichen des HTML-Attributs kein Problem dar.

Sie können nach dem echo-Befehl zwar einfache oder doppelte Anführungszeichen verwenden, aber es ist besser, sich für eine Variante zu entscheiden und diese beizubehalten. In diesem Buch verwenden wir meist einfache Anführungszeichen, damit der Inhalt HTML-Attribute enthalten kann, so wie hier gezeigt.

Probieren Sie aus: Ändern Sie in Schritt 1 den Namen Ivy in Ihren eigenen Namen ab und speichern Sie die Datei. Wenn Sie die Seite aktualisieren, verwendet die Begrüßung Ihren Namen.

KOMMENTARE

Es gehört zur guten Praxis, Ihren PHP-Code mit Kommentaren zu versehen, die seine Funktion beschreiben. So erinnern Sie sich besser, was Ihr Code tut, wenn Sie nach einiger Zeit zurückkehren, und auch anderen hilft es, Ihren Code zu verstehen.

Einzelige Kommentare werden eingeleitet durch:

- zwei Schrägstriche // oder
- ein einzelnes Rauten- (oder Hashtag-)Symbol #

Diese Zeichen weisen den PHP-Interpreter an, den restlichen PHP-Code auf dieser Zeile zu ignorieren, bis er auf einen abschließenden ?> PHP-Tag trifft.

```
echo "Welcome"; // Begrüßung anzeigen  
echo "Welcome"; # Begrüßung anzeigen
```

EINZEILIGER KOMMENTAR

Mehrzeilige Kommentare erstrecken sich über mehrere Zeilen der PHP-Datei. Darin können Sie detailliertere Beschreibungen und/oder Anmerkungen zu Ihrem Code unterbringen.

```
echo "Welcome";  
/*
```

Nach Begrüßungsnachricht:

- Profilbild neben Mitgliedername hinzufügen
- Beides als Link zur Profilseite des Nutzers

Ein Schrägstrich und ein Sternchen /* weisen den PHP-Interpreter an, alles Weitere zu ignorieren, bis er auf ein Sternchen gefolgt von einem Schrägstrich stößt */.

MEHRZEILIGER KOMMENTAR

KOMMENTARE ZUM CODE HINZUFÜGEN

PHP

section_a/intro/comments.php

```
<?php
/*
Diese Seite zeigt den Mitgliedsnamen
und Details zu einem aktuellen Angebot
*/
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Adding Comments to Your Code</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2><?php echo 'Welcome Ivy'; //Zeige Name ?></h2>
    <?php echo '<p class="offer">Offer: 20% off</p>' ?>
  </body>
</html>
```

ERGEBNIS



Dieses Beispiel ist dem vorherigen sehr ähnlich, ergänzt den Code jedoch um Kommentare.

1. Die Seite beginnt mit einem mehrzeiligen Kommentar, der den Zweck des Codes beschreibt.

2. Nach der Begrüßung erläutert ein einzeiliger Kommentar, was hier angezeigt wird.

Die Kommentare werden nicht in den HTML-Code eingefügt, der an den Browser gesendet wird; sie erscheinen nur im PHP-Code.

Probieren Sie aus: Fügen Sie in Schritt 1 eine weitere Textzeile in den Kommentar ein.

Probieren Sie aus: Ändern Sie in Schritt 2 die doppelten Schrägstriche in ein # (das Rauten-/Hashtag-Symbol).

Hinweis: In diesem Buch finden Sie zahlreiche Kommentare, die beschreiben, was die einzelnen Code-Zeilen in den Beispielen tun. Erfahrene Programmierer kommentieren nur selten jede einzelne Zeile.

KAPITEL IN TEIL A

GRUNDLAGEN DER PROGRAMMIERUNG

1

VARIABLEN, AUSDRÜCKE & OPERATOREN

Wenn eine PHP-Seite eine Aufgabe ausführt, für die sie konzipiert wurde, kann sie bei jedem neuen Durchlauf unterschiedliche Werte verwenden. Daher ist es wichtig, diese Daten im Code mithilfe von Variablen repräsentieren zu können. Sie lernen zudem, wie Sie Ausdrücke und Operatoren einsetzen, um mit diesen Werten zu arbeiten.

2

KONTROLLSTRUKTUREN

Eine PHP-Seite führt nicht immer die gleichen Code-Zeilen in der gleichen Reihenfolge aus. Mithilfe von Kontrollstrukturen können Sie Regeln schreiben, nach denen der PHP-Interpreter entscheidet, welchen Code er als Nächstes ausführen soll.

3

FUNKTIONEN

Alle Einzelanweisungen, die zur Ausführung einer Aufgabe erforderlich sind, lassen sich in einer Funktion zusammenfassen. Dies hilft nicht nur, Ihren Code besser zu strukturieren, sondern es erspart Ihnen auch, dieselben Anweisungen immer aufs Neue zu wiederholen, wenn die Seite eine bestimmte Aufgabe mehrfach durchführen muss.

4

KLASSEN & OBJEKTE

Konzepte wie die Mitglieder, die verkauften Produkte und die angezeigten Artikel einer Webseite werden in Code gegossen. Programmierer verwenden Klassen und Objekte, um den Code zusammenzufassen, der jedes dieser verschiedenen Konzepte repräsentiert.

1

VARIABLEN, AUSDRÜCKE & OPERATOREN

Dieses Kapitel zeigt, wie Daten, die sich bei jeder Anforderung einer PHP-Seite ändern können, in Variablen gespeichert werden und wie Ausdrücke und Operatoren mit Variablenwerten umgehen.

Variablen stellen mit einem Namen einen Wert dar, der sich bei jedem Aufruf einer PHP-Seite ändern kann:

- Der **Name** beschreibt die Art der Daten, die die Variable enthält.
- Der **Wert** ist der Wert, den die Variable beim Aufruf der Seite enthalten soll.
Sobald die Seite fertig abgearbeitet und der HTML-Code an den Browser zurückgesendet wurde, vergisst der PHP-Interpreter die Variable wieder (und sie kann dann beim nächsten Aufruf der Seite einen anderen Wert enthalten).

PHP unterscheidet zwischen verschiedenen Arten von Werten, die Sie in einer Variablen speichern können (z.B. Text und Zahlen); diese werden als **Datentypen** bezeichnet:

- Ein Text wird als **Zeichenkette (String)** bezeichnet.
- Eine ganze Zahl heißt **Integer**.
- Ein Dezimalbruch entspricht dem Datentyp **Float**.
- Ein Wert, der wahr oder falsch (`true` oder `false`) sein kann, ist ein **boolescher Wert (Boolean)**.
- Eine Reihe zusammenhängender Namen und Werte lässt sich in einem **Array** speichern.

Wenn Sie sich erst einmal mit Variablen vertraut gemacht haben, werden Sie sehen, dass Ausdrücke aus mehreren Werten einen einzigen Wert bilden können. So lässt sich z.B. Text aus zwei Variablen zu einem Satz zusammenfügen oder eine in einer Variablen gespeicherte Zahl mit einer Zahl aus einer anderen Variablen multiplizieren.

Ausdrücke verwenden **Operatoren**, um einen einzelnen Wert zu erzeugen. Der Operator `+` wird beispielsweise zur Addition zweier Werte verwendet, der Operator `-` zur Subtraktion eines Werts von einem anderen.



VARIABLEN

Variablen speichern Daten, die sich bei jedem Aufruf einer PHP-Seite verändern (variieren) können. Sie stellen also einen veränderlichen Wert mit einem feststehenden Namen dar.

Um eine Variable zu erstellen und einen Wert darin zu speichern, benötigen Sie:

- einen **Variablennamen**, der mit einem Dollarzeichen beginnen muss, gefolgt von einem oder mehreren Wörtern, die die Art der von der Variablen zu speichernden Informationen beschreiben.
- ein **Gleichheitszeichen**, das auch als Zuweisungsoperator bezeichnet wird, weil es dem Variablennamen einen Wert zuweist.
- den **Wert**, der in der Variablen abgelegt werden soll.

Wenn die Variable Text enthält, muss dieser in Anführungszeichen gesetzt werden. Sie können einfache oder doppelte Anführungszeichen verwenden, aber sie müssen zusammenpassen. (Beginnen Sie zum Beispiel nicht mit einem einfachen Anführungszeichen und schließen dann mit einem doppelten Anführungszeichen ab.)

Wenn die Variable eine Zahl oder einen booleschen Wert (`true` oder `false`) enthält, wird dieser nicht in Anführungszeichen gesetzt.

Die Erstellung einer Variablen bezeichnen Programmierer als **Variablen-deklaration**. Nach der Deklaration erfolgt dann üblicherweise die **Zuweisung** eines Werts.

NAME	WERT
<code>\$name</code>	'Ivy'
<code>\$price</code>	5;
ZUWEISUNGSPERATOR	

Sobald eine Variable deklariert und ihr ein Wert zugewiesen wurde, können Sie den Variablennamen im PHP-Code überall dort nutzen, wo Sie den aktuellen Wert der Variablen benötigen.

Wenn der PHP-Interpreter auf einen Variablennamen stößt, ersetzt er den Namen durch den darin enthaltenen Wert. Im Folgenden zeigen wir den in der oben gezeigten Variablen `$name` gespeicherten Wert mit dem Befehl `echo` an.

ZEIGE AN	\$name;
echo	WERT IN VARIABLE

VARIABLEN ERSTELLEN UND DARAUF ZUGREIFEN

PHP

section_a/c01/variables.php

```
<?php  
① $name  = 'Ivy';  
② $price = 5;  
?>  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Variables</title>  
    <link rel="stylesheet" href="css/styles.css">  
  </head>  
  <body>  
    <h1>The Candy Store</h1>  
    ③ <h2>Welcome <?php echo $name; ?></h2>  
    <p>The cost of your candy is  
      ④   $<?php echo $price; ?> per pack.</p>  
  </body>  
</html>
```

ERGEBNIS



In diesem Beispiel werden am Anfang der Seite zwei Variablen erstellt und mit Werten gefüllt:

1. \$name enthält den Namen des aktuellen Website-Besuchers. Da es sich um Text handelt, steht dieser in Anführungszeichen.
 2. \$price enthält den Preis für eine einzelne Süßigkeit. Da es sich um eine Zahl handelt, steht der Wert nicht in Anführungszeichen.
- Als Nächstes sehen Sie den HTML-Code, der an den Browser zurückgesendet wird. Darin wird:
3. der Benutzername mit dem Befehl echo in die Seite geschrieben
 4. der Süßwarenpreis in die Seite geschrieben

Probieren Sie es: Ändern Sie in Schritt 1 den Wert der Variablen \$name so ab, dass er Ihren Namen enthält. Speichern Sie die Datei und aktualisieren Sie dann die Seite in Ihrem Browser. Sie sehen daraufhin Ihren Namen.

Probieren Sie es: Ändern Sie in Schritt 2 den Preis auf 2. Speichern Sie die Datei und aktualisieren Sie dann die Seite. Nun wird der neue Preis angezeigt.

In diesem Kapitel weisen wir den Variablen direkt im PHP-Code Werte zu. In künftigen Kapiteln stammen die Werte, die in die Variablen geschrieben werden, aus von den Besuchern abgeschickten HTML-Formularen, aus URL-Daten und aus Datenbanken.

ZUR BENENNUNG VON VARIABLEN

Ein Variablenname sollte die in der Variablen gespeicherten Daten beschreiben. Halten Sie folgende Regeln ein, um einen Variablenamen zu erstellen.

1

2

3

Beginnen Sie mit einem Dollarzeichen (\$).

- \$greeting
- greeting

Darauf folgt ein Buchstabe oder ein Unterstrich (keine Zahl).

- \$greeting
- \$2_greeting

Wenn Sie Variablennamen mit beschreibendem Charakter verwenden, ist Ihr Code leichter zu verstehen und nachzuvollziehen.

Wenn Sie mehrere Wörter verwenden, um die in einer Variablen enthaltenen Daten zu beschreiben, werden diese üblicherweise durch einen Unterstrich getrennt.

Die Groß- und Kleinschreibung ist bei der Namensgebung zu beachten, sodass \$Score und \$score zwei verschiedene Namen bilden würden. Sie sollten es jedoch generell vermeiden, zwei Variablen zu erstellen, die dasselbe Wort in unterschiedlicher Schreibweise verwenden, da Sie damit andere Personen, die Ihren Code lesen, verwirren könnten.

Nutzen Sie dann eine beliebige Kombination aus Buchstaben von A bis Z (Groß- und Kleinbuchstaben), Zahlen und Unterstrichen. (Bindestriche oder Punkte sind nicht erlaubt.)

- \$greeting_2
- \$greeting-2
- \$greeting.2

Hinweis: \$this hat eine besondere Bedeutung. Verwenden Sie es nicht als Variablenname.

- \$this

Aus technischen Gesichtspunkten können Sie Zeichen aus verschiedenen Zeichensätzen verwenden (z.B. chinesische oder kyrillische Zeichen), aber es wird häufig als gängige Praxis angesehen, nur die Buchstaben A-z, Zahlen und Unterstriche zu verwenden (da die Unterstützung anderer Zeichen mit einigen Schwierigkeiten verbunden ist).

SKALARE (EINFACHE) DATENTYPEN

PHP unterscheidet zwischen drei skalaren Datentypen, die Text, Zahlen und boolesche Werte enthalten.

STRING-DATENTYP

Programmierer bezeichnen ein Textstück als Zeichenkette oder **String**. Der String-Datentyp kann aus Buchstaben, Zahlen und anderen Zeichen bestehen, die jedoch zur Darstellung von Text verwendet werden.

`$name = 'Ivy';`

Zeichenketten werden immer von einfachen oder doppelten Anführungszeichen umgeben. Das öffnende Anführungszeichen muss mit dem schließenden Anführungszeichen übereinstimmen.

✓ `$name = 'Ivy';`

✓ `$name = "Ivy";`

✗ `$name = "Ivy';`

✗ `$name = 'Ivy";`

NUMERISCHE DATENTYPEN

Numerische Datentypen ermöglichen es, mathematische Operationen mit den darin gespeicherten Werten durchzuführen, wie etwa Additionen oder Multiplikationen.

`$price = 5;`

Zahlen werden nicht in Anführungszeichen gesetzt. Wenn Sie Zahlen in Anführungszeichen setzen, werden sie statt als Zahlen als Zeichenketten behandelt.

PHP kennt zwei numerische Datentypen:

`int` steht für Integer, also für ganze Zahlen (z. B. 275).

`float` enthält Fließkomma-Zahlen, die Dezimalbrüche darstellen (z. B. 2,75).

BOOLESCHE DATENTYPEN

Der boolesche Datentyp kann nur einen von zwei Werten annehmen: `true` oder `false`. Diese Werte sind in den meisten Programmiersprachen üblich.

`$logged_in = true;`

`true` und `false` sollten in Kleinbuchstaben geschrieben und nicht in Anführungszeichen gesetzt werden. Auf den ersten Blick wirken boolesche Werte vielleicht abstrakt, aber mit `true` oder `false` lassen sich viele Dinge ausdrücken, wie zum Beispiel:

- Ist jemand eingeloggt?
- Hat die Person den allgemeinen Geschäftsbedingungen zugestimmt?
- Ist ein Produkt für den kostenlosen Versand qualifiziert?

NULL-DATENTYP

PHP kennt auch den Datentyp `null`. Dieser kann nur den Wert `null` haben. Er zeigt an, dass für eine Variable noch kein Wert definiert wurde.

MIT DATENTYPEN JONGLIEREN

Auf den Seiten 60 und 61 sehen Sie, wie der PHP-Interpreter verschiedene Datentypen ineinander umwandeln kann (z. B. einen String in einen Zahlenwert).

EINEN VARIABLENWERT AKTUALISIEREN

Sie können den in einer Variablen abgelegten Wert ändern oder überschreiben, indem Sie ihr einen neuen Wert zuweisen. Das erfolgt auf die gleiche Weise, wie Sie der Variablen bei der Erstellung einen Wert zuweisen.

1. Die Variable \$name wird **initialisiert**. Das bedeutet, dass sie deklariert und ihr ein Anfangswert zugewiesen wird. Dieser wird verwendet, sofern die Variable später auf der Seite nicht aktualisiert wird.

Der Anfangswert ist Guest; da es sich um Text handelt, wird er in Anführungszeichen geschrieben.

2. Der Variablen \$name wird nachfolgend der neue Wert Ivy zugewiesen.

3. Die Variable \$price enthält den Preis für eine einzelne Packung Süßigkeiten.

Als Nächstes sehen Sie den HTML-Code, der an den Browser zurückgesendet wird. Darin wird:

4. der Name mit dem Befehl echo in die Seite geschrieben.

Angezeigt wird der aktualisierte Wert, der der Variablen \$name in Schritt 2 zugewiesen wurde.

5. der Süßigkeitenpreis auf die Seite geschrieben.

section_a/c01/updating-variables.php

PHP

```
<?php  
① $name = 'Guest';  
② $name = 'Ivy';  
③ $price = 5;  
?>  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Updating Variables</title>  
    <link rel="stylesheet" href="css/styles.css">  
  </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2>Welcome <?php echo $name; ?></h2>  
    <p>The cost of your candy is  
      <?php echo $price; ?> per pack.</p>  
  </body>  
</html>
```

ERGEBNIS



Probieren Sie aus: Ändern Sie in Schritt 2 den Wert der Variablen \$name so, dass er Ihren Namen enthält. Speichern Sie die Datei und aktualisieren Sie dann die Seite in Ihrem Browser. Nun wird Ihr Name angezeigt.

Probieren Sie aus: Fügen Sie nach Schritt 2 eine neue Zeile ein und geben Sie der Variablen \$name einen anderen Namen. Speichern Sie die Datei und aktualisieren Sie die Seite. Sie sehen nun den neuen Namen.

ARRAYS

Eine Variable kann auch ein **Array** oder **Datenfeld** enthalten, in dem eine Reihe zusammengehöriger Werte gespeichert ist. Arrays werden auch zu den **zusammengesetzten Datentypen** gezählt, da sie mehr als einen Wert speichern können.

Ein Array ist wie ein Behälter, der eine Reihe zusammengehöriger Variablen enthält. Jeder Eintrag in einem Array wird als **Element** bezeichnet. Ebenso wie eine Variable einen Namen für die Darstellung eines Werts verwendet, besitzt auch jedes Element in einem Array:

- einen **Schlüssel**, der sich genau wie ein Variablenname verhält
- einen **Wert**, also die zum Namen gehörigen Daten

PHP kennt zwei Arten von Arrays:

- In **assoziativen Arrays** bildet ein Name, der die darunter abgelegten Daten beschreibt, den Schlüssel für jedes Element.
- In **indizierten Arrays** bestehen die Elementschlüsse aus fortlaufenden Zahlen, den sogenannten **Indexwerten**.

ASSOZIATIVES ARRAY

Das nachstehende Array dient der Speicherung von Daten, die für ein Mitglied einer Website stehen. Bei jeder Verwendung des Arrays bleiben die Schlüsselnamen (die die in den einzelnen Elementen des Arrays gespeicherten Daten beschreiben) gleich.

SCHLÜSSEL	WERT
\$member	
name	=> Ivy
age	=> 32
country	=> Italy

In diesen beiden Beispielen ist jeder im Array gespeicherte Wert ein skalarer Datentyp (ein einzelnes Datenelement).

Auf Seite 44 finden Sie Beispiele für Arrays, deren Elemente selbst auch wieder ein weiteres Array enthalten.

INDIZIERTES ARRAY (STANDARD-ARRAY)

Das nachstehende Array dient zur Erfassung einer Einkaufsliste. Derartige Listen können bei jeder Nutzung eine andere Anzahl von Elementen enthalten. Der Schlüssel nutzt keinen Namen zur Beschreibung der einzelnen Listenelemente, sondern einen aufsteigenden Indexwert (ganzahlig und bei 0 beginnend).

SCHLÜSSEL	WERT
\$shopping_list	=
0	=> bread
1	=> cheese
2	=> milk

Hinweis: Die Indexwerte beginnen bei 0, nicht bei 1. Das erste Listenelement hat den Index 0, das zweite Element hat den Index 1 und so weiter. Mit der Indexnummer wird häufig die Reihenfolge der Listenelemente beschrieben.

ASSOZIATIVE ARRAYS

Um ein assoziatives Array zu erstellen, weisen Sie jedem Element (oder Eintrag) im Array einen **Schlüssel** zu, der die darin enthaltenen Daten beschreibt.

Um ein assoziatives Array in einer Variablen zu speichern, verwenden Sie:

- einen Variablenamen, der alle im Array enthaltenen Werte beschreibt
- den Zuweisungsoperator
- eckige Klammern zur Erstellung des Arrays

Innerhalb der eckigen Klammern verwenden Sie:

- den Schlüsselnamen in Anführungszeichen
- den Doppelpfeil-Operator =>
- den Wert für dieses Element (Zeichenketten stehen in Anführungszeichen, Zahlen und boolesche Werte nicht)
- ein Komma nach jedem Element

```
VARIABLE      ARRAY ANLEGEN
|             |
$member = [
    'name'     => 'Ivy',
    'age'       => 32,
    'country'   => 'Italy',
];
SCHLÜSSEL    OPERATOR    WERT
```

Ein assoziatives Array kann auch mit der unten dargestellten Syntax erstellt werden, wobei auf das Wort array dann runde (anstelle von eckigen) Klammern folgen.

```
$member = array(
    'name'     => 'Ivy',
    'age'       => 32,
    'country'   => 'Italy',
);
```

Um auf ein Element eines assoziativen Arrays zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern und Anführungszeichen
- den Schlüssel des abzurufenden Elements

```
VARIABLE      SCHLÜSSEL
|             |
$member['name'];
```

ASSOZIATIVE ARRAYS ANLEGEN UND DARAUF ZUGREIFEN

PHP

section_a/c01/associative-arrays.php

```
<?php  
① $nutrition = [  
    'fat'  => 16,  
    'sugar' => 51,  
    'salt'  => 6.3,  
];  
?  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Nutrition (per 100g)</h2>  
        <p>Fat:  <?php echo $nutrition['fat']; ?>%</p>  
        <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>  
        <p>Salt:   <?php echo $nutrition['salt']; ?>%</p>  
    </body>  
</html>
```

ERGEBNIS



Probieren Sie aus: Fügen Sie in Schritt 1 ein weiteres Element in das Array ein. Verwenden Sie den Schlüssel protein und weisen Sie ihm einen Wert von 2.6 zu. Lassen Sie dann in Schritt 2 den Proteinwert auf der Seite anzeigen.

1. In diesem Beispiel erstellen wir ein assoziatives Array und speichern es in der Variablen \$nutrition. Das Array wird in eckigen Klammern erstellt. Es besteht aus drei Elementen (jedes Element umfasst ein Schlüssel/Wert-Paar). Der Operator => ordnet den Schlüsseln die einzelnen Werte zu.

2. Um im Array gespeicherte Daten anzuzeigen, verwenden Sie:

- den echo-Befehl, der dafür sorgt, dass der nachfolgende Wert auf der Webseite ausgegeben wird
- gefolgt von dem Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern und Anführungszeichen, die den Schlüsselnamen enthalten, auf dessen Wert Sie zugreifen möchten

Um beispielsweise den Zuckergehalt auf der Seite auszugeben, verwenden Sie:

```
echo $nutrition  
['sugar'];
```

Probieren Sie aus: Ändern Sie in Schritt 1 die Werte des Arrays. Geben Sie dem Schlüssel:

- fat einen Wert von 42
- sugar einen Wert von 60
- salt einen Wert von 3.5

Speichern und aktualisieren Sie die Seite, um die aktualisierten Werte zu sehen.

INDIZIERTE ARRAYS

Wenn bei der Deklaration eines Arrays nicht für jedes Element ein Schlüssel angegeben wird, weist der PHP-Interpreter ihm eine Zahl zu, die Indexnummer. Indexnummern beginnen bei null (0), nicht bei eins (1).

Um ein indiziertes Array in einer Variablen zu speichern, verwenden Sie:

- einen Variablenamen, der alle im Array enthaltenen Werte beschreibt
- den Zuweisungsoperator
- eckige Klammern zur Erstellung des Arrays

Innerhalb der eckigen Klammern verwenden Sie:

- die Liste der Werte, die im Array gespeichert werden sollen (Zeichenketten stehen in Anführungszeichen, Zahlen und boolesche Werte nicht)
 - ein Komma nach jedem Wert
- Jedes Element erhält dabei eine Indexnummer.

VARIABLE	ZUWEISUNGS- OPERATOR	WERTE
\$shopping_list	=	['bread', 'cheese', 'milk',];

Oben würde bread die Indexnummer 0, cheese die 1 und milk die 2 erhalten. Mit Indexnummern wird oft die Reihenfolge der im Datenfeld enthaltenen Elemente angegeben.

Ein indiziertes Array kann auch mit der unten dargestellten Syntax erstellt werden, wobei auf das Wort array dann runde (anstelle von eckigen) Klammern folgen.

```
$shopping_list = array('bread',
                      'cheese',
                      'milk');
```

Neue Werte können dem Array in derselben oder in einer neuen Zeile (so wie oben gezeigt) zugewiesen werden.

Um auf die Daten eines indizierten Arrays zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern (keine Anführungszeichen)
- dem Indexwert des Elements, auf das Sie zugreifen möchten (in den eckigen Klammern)

Der nachfolgende Code greift auf das dritte Array-Element zu, in diesem Beispiel als auf den Wert milk.

VARIABLE	INDEXNUMMER
\$shopping_list	[2];

INDIZIERTE ARRAYS ANLEGEN UND DARAUF ZUGREIFEN

PHP

section_a/c01/indexed-arrays.php

```
<?php  
① [ $best_sellers = ['Chocolate', 'Mints', 'Fudge',  
    'Bubble gum', 'Toffee', 'Jelly beans',];  
?  
<!DOCTYPE html>  
<html>  
  <head> ... </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2>Best Sellers</h2>  
    <ul>  
      <li><?php echo $best_sellers[0]; ?></li>  
      <li><?php echo $best_sellers[1]; ?></li>  
      <li><?php echo $best_sellers[2]; ?></li>  
    </ul>  
  </body>  
</html>
```

ERGEBNIS



1. In diesem Beispiel erstellen wir zunächst die Variable `$best_sellers`. Als Variablenwert enthält sie ein Array mit einer Liste der meistverkauften Artikel auf der Website.

Dieses Array wird mittels eckiger Klammern erstellt, und seine Elemente stehen innerhalb dieser Klammern. Da es sich bei den Elementen im Array um Text handelt, stehen sie zudem in Anführungszeichen. (Zahlen und boolesche Werte gehören nicht in Anführungszeichen.) Auf jedes Element folgt ein Komma.

2. Hier werden die drei meistverkauften Artikel auf die Seite geschrieben:

- der `echo`-Befehl sorgt für die Ausgabe des nachfolgenden Werts
- gefolgt wird er vom Namen der Variablen, die das Array enthält
- danach steht die Indexnummer des abzurufenden Elements in eckigen Klammern. Denken Sie daran, dass Indexnummern bei 0 beginnen, nicht bei 1.

Probieren Sie aus: Fügen Sie in Schritt 1 hinter Fudge noch Licorice in die Auflistung ein. In Schritt 2 lassen Sie auch noch den 4. und 5. Artikel aus dem Array anzeigen.

ARRAYS AKTUALISIEREN

Nachdem Sie ein Array erstellt haben, können Sie ihm neue Elemente hinzufügen oder den Wert eines bestehenden Elements aktualisieren.

Um einen in einem assoziativen Array abgelegten Wert zu aktualisieren, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern
- nun den Schlüsselnamen in Anführungszeichen
- einen Zuweisungsoperator
- den neuen Wert, den es enthalten soll

`$member['name'] = 'Tom';`

VARIABLE SCHLÜSSEL NEUER WERT

Um einen in einem indizierten Array abgelegten Wert zu aktualisieren, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern
- die Indexnummer (ohne Anführungszeichen)
- einen Zuweisungsoperator
- den neuen Wert, den es enthalten soll

`$shopping_list[2] = 'butter';`

VARIABLE INDEX-
NUMMER NEUER WERT

Um einem assoziativen Array ein neues Element hinzuzufügen, gehen Sie genauso vor wie oben beschrieben, nutzen dabei aber einen neuen Schlüsselnamen (keinen, der bereits im Array verwendet wurde).

Die Anführungszeichen umschließen den Schlüsselnamen, wenn es sich dabei um eine Zeichenkette handelt, da Anführungszeichen einen String-Datentyp anzeigen.

Wie man Elemente zu indizierten Arrays hinzufügt, erfahren Sie auf Seite 220. Der Ablauf ist etwas anders, da Sie die Position des neuen Elements im Array bestimmen können.

Indexnummern stehen nicht in Anführungszeichen, da numerische Datentypen nicht in Anführungszeichen geschrieben werden.

WELCHER ARRAY-TYP FÜR WELCHE ANWENDUNG?

Assoziative Arrays sind am besten geeignet, wenn Sie:

- genau wissen, welche Informationen das Array enthalten soll. Dies ist erforderlich, um für jedes Element einen Schlüsselnamen zu vergeben.
- einzelne Daten unter Verwendung eines Schlüsselnamens abrufen müssen.

Indizierte Arrays sind hilfreich, wenn Sie:

- nicht wissen, wie viele Daten in dem Array gespeichert werden sollen. (Der Index wird stetig erweitert, je mehr Elemente Sie der Liste hinzufügen.)
- eine Reihe von Werten in einer bestimmten Reihenfolge abspeichern möchten.

IN ARRAYS GESPEICHERTE WERTE VERÄNDERN

PHP

section_a/c01/updating-arrays.php

```
<?php  
① $nutrition = [  
    'fat'  => 38,  
    'sugar' => 51,  
    'salt'  => 0.25,  
];  
② $nutrition['fat']  = 36;  
③ $nutrition['fiber'] = 2.1;  
?  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Nutrition (per 100g)</h2>  
        <p>Fat:  <?php echo $nutrition['fat']; ?>%</p>  
        <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>  
        <p>Salt:  <?php echo $nutrition['salt']; ?>%</p>  
        <p>Fiber: <?php echo $nutrition['fiber']; ?>%</p>  
    </body>  
</html>
```

1. In diesem Beispiel wird zunächst ein Array in der Variablen \$nutrition abgelegt.

Die Schlüssel und Werte, die die einzelnen Array-Elemente bilden, müssen nicht unbedingt auf einer neuen Zeile stehen (so wie hier gezeigt), allerdings erhöht diese Praxis die Lesbarkeit.

2. Der Wert für den Fettgehalt wird von 38 auf 36 aktualisiert.
3. Ein neues Element wird zum Array hinzugefügt. Der Schlüssel heißt fiber und sein Wert ist 2.1.

4. Die Werte im Array werden auf der Seite ausgegeben.

Probieren Sie aus: Fügen Sie nach Schritt 3 einen weiteren Schlüssel für Eiweiß (protein) hinzu und weisen Sie ihm den Wert 7.3 zu.

ERGEBNIS



ARRAYS IN EINEM ARRAY SPEICHERN

Der Wert eines jeden Elements innerhalb eines Arrays kann ein weiteres Array sein. Enthält jedes Array-Element ein weiteres Array, spricht man von einem **mehrdimensionalen Array**. Dieses eignet sich für die Darstellung von Daten, wie sie etwa in Tabellen vorkommen.

In einigen Fällen müssen Sie einen zusammenhängenden Wertesatz in einem Element eines Arrays speichern (z. B. bei der Darstellung von Daten, die sonst üblicherweise in Tabellen zu finden sind). Sehen Sie sich die rechte Tabelle mit drei Personen, ihrem Alter und ihren jeweiligen Aufenthaltsorten an.

Jede Zeile dieser Tabelle (jede Person) lässt sich durch ein Element eines indizierten Arrays darstellen. Jedes dieser Elemente kann dann wiederum ein assoziatives Array enthalten, in dem der Name, das Alter und das Land der jeweiligen Person gespeichert sind.

```
$members = [  
    ['name' => 'Ivy', 'age' => 32, 'country' => 'UK'],  
    ['name' => 'Emi', 'age' => 24, 'country' => 'Japan'],  
    ['name' => 'Luke', 'age' => 47, 'country' => 'USA'],  
];
```

Um das Array mit den Daten über Emi zu erhalten, verwenden Sie:

- den Namen der Variablen, die das indizierte Array enthält
- die Indexnummer des Elements, auf das Sie zugreifen möchten in eckigen Klammern (denken Sie daran, dass die Zählung bei 0 beginnt und dass Zahlen nicht in Anführungszeichen gesetzt werden).

```
$members[1];
```

NAME	ALTER	LAND
Ivy	32	UK
Emi	24	Japan
Luke	47	USA

Die Indexnummern für das indizierte Array werden automatisch vom PHP-Interpreter zugewiesen. Das Komma nach jedem assoziativen Array zeigt das Ende des Werts für dieses Element an.

Um das Alter von Luke abzurufen, verwenden Sie:

- den Namen der Variablen, die das indizierte Array enthält
- die Indexnummer des Elements, das die Informationen über Luke enthält in eckigen Klammern
- den Schlüssel des Elements, auf das Sie im Array über Luke zugreifen möchten in einem zweiten Satz eckiger Klammern (der Schlüssel ist eine Zeichenkette und gehört daher in Anführungszeichen gesetzt)

```
$members[2]['age'];
```

MEHRDIMENSIONALE ARRAYS

PHP

section_a/c01/multidimensional-arrays.php

```
<?php  
$offers = [  
    ['name' => 'Toffee', 'price' => 5, 'stock' => 120,],  
    ['name' => 'Mints', 'price' => 3, 'stock' => 66,],  
    ['name' => 'Fudge', 'price' => 4, 'stock' => 97,],  
];  
?>  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Offers</h2>  
        <p><?php echo $offers[0]['name']; ?> -  
            $<?php echo $offers[0]['price']; ?> </p>  
        <p><?php echo $offers[1]['name']; ?> -  
            $<?php echo $offers[1]['price']; ?> </p>  
        <p><?php echo $offers[2]['name']; ?> -  
            $<?php echo $offers[2]['price']; ?> </p>  
    </body>  
</html>
```

ERGEBNIS



1. Dieses Beispiel beginnt mit der Hinterlegung eines indizierten Arrays in der Variablen \$offers.

Jedes Element im Array enthält ein assoziatives Array mit dem Namen, Preis und Lagerbestand eines angebotenen Artikels.

2. Der Name des ersten Produkts wird ausgegeben. (Die Indexnummer des ersten Produkts ist 0.)
3. Der Preis des ersten Produkts wird ausgegeben.

4. Der Name und der Preis des zweiten Produkts werden ausgegeben.

5. Der Name und der Preis des dritten Produkts werden ausgegeben.

Probieren Sie es: Fügen Sie in Schritt 1 ein weiteres Produkt mit dem Namen Chocolate zum Array hinzu. Setzen Sie den Preis auf 2 und den Lagerbestand auf 83. Geben Sie dann nach Schritt 5 den Namen und den Preis des neu hinzugefügten Produkts aus.

Im nächsten Kapitel lernen Sie, wie Sie mithilfe einer Schleife den Namen und den Preis aller im Array \$offers enthaltenen Produkte ausgeben können, und zwar unabhängig von der Anzahl der darin enthaltenen Produkte.

KURZFORM FÜR ECHO

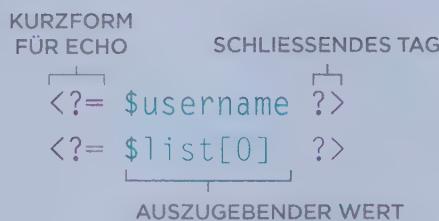
Wenn ein PHP-Block nur dazu dient, einen Wert in den Browser zu schreiben, können Sie anstelle von `<?php echo ?>` eine Abkürzung verwenden.

Anstatt `<?php echo $name; ?>` zu schreiben, können Sie die Abkürzung `<?= $name ?>` verwenden.

Dies ist die einzige Situation, in der Sie nicht das vollständige öffnende `<?php`-Tag brauchen.

Das können Sie weglassen:

- Die Buchstaben `php` im öffnenden Tag
- Den `echo`-Befehl
- Den Strichpunkt vor dem schließenden Tag



In vielen der Beispiele in den ersten Buchkapiteln wird deutlich, dass jede PHP-Datei aus zwei Teilen besteht:

- Zuerst speichert der PHP-Code Werte in Variablen oder Arrays. (Er kann auch bestimmte Aktionen mit den darin enthaltenen Daten durchführen.)
- Danach folgt der HTML-Code, der an den Browser zurückgesendet wird. Dieser zweite Teil der Seite gibt mit der oben gezeigten Kurzsyntax Werte aus, die in Variablen gespeichert wurden.

Wenn Sie zu Beginn jeder Seite die auf der Seite anzuzeigenden Werte erzeugen und in Variablen speichern, können Sie eine klare Trennung zwischen dem auf dem Server ausgeführten PHP-Code und dem HTML-Code, der letztlich im Browser dargestellt wird, herstellen.

Im zweiten Teil der Datei, in dem die HTML-Seite erstellt wird, sollte so wenig PHP-Code wie möglich vorkommen. In unseren ersten Beispielen schreibt der PHP-Code in diesem Teil der Seite lediglich in Variablen gespeicherte Werte in die HTML-Seite.

DIE KURZFORM FÜR ECHO VERWENDEN

PHP

Übung 1: echo shorthand / section_a/c01/echo-shorthand.php

```
<?php  
① $name      = 'Ivy';  
② $favorites = ['Chocolate', 'Toffee', 'Fudge',];  
?>  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Echo Shorthand</title>  
    <link rel="stylesheet" href="css/styles.css">  
  </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2>Welcome <?= $name ?></h2>  
    <p>Your favorite type of candy is:  
      <?= $favorites[0] ?>.</p>  
  </body>  
</html>
```

ERGEBNIS



In diesem Beispiel werden zwei verschiedene Variablen erstellt und am Anfang der Seite mit Werten gefüllt, bevor der HTML-Code beginnt:

1. `$name` enthält den Namen eines Website-Nutzers. Da es sich um Text handelt, wird er in Anführungszeichen gesetzt.
2. `$favorites` enthält eine Reihe von Lieblingssüßigkeiten dieses Mitglieds.
3. Der Name wird mithilfe der Kurzschreibweise des echo-Befehls in die Seite geschrieben.
4. Die Lieblingssüßigkeit des Mitglieds wird mit der Kurzschreibweise des echo-Befehls in die Seite geschrieben.

Probieren Sie es: Ändern Sie in Schritt 1 den in der Variablen `$name` gespeicherten Wert in Ihren eigenen Namen. In Schritt 2 setzen Sie Ihre Lieblingssüßigkeit an den Anfang des Arrays. Speichern Sie die Datei und aktualisieren Sie die Seite in Ihrem Browser. Daraufhin sehen Sie, wie sich der Inhalt ändert.

AUSDRÜCKE & OPERATOREN

Häufig wird aus zwei (oder mehr) Werten ein neuer Wert erzeugt. **Ausdrücke** bestehen aus einem oder mehreren Konstrukten, die zur Ausgabe eines einzelnen Werts führen. Ausdrücke nutzen hierfür **Operatoren**.

Bei den Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) werden zwei Werte zu einem neuen Wert verrechnet. Der folgende Ausdruck multipliziert die Zahl 3 mit der Zahl 5 und erzeugt dabei den Wert 15:

```
3 * 5
```

Ausdrücke ergeben also einen einzelnen Wert. Nachfolgend wird der neu erzeugte Wert in der Variablen \$total gespeichert:

```
$total = 3 * 5;
```

Die Zeichen + - * / = werden als **Operatoren** bezeichnet.

Mit dem String-Operator (**Verknüpfungsoperator**) können Sie mindestens zwei Strings zu einem einzelnen längeren Text zusammenfügen. Der folgende Ausdruck verknüpft die Werte »Hi« und »Ivy« zu einer einzelnen Zeichenkette.

```
$greeting = 'Hi' . 'Ivy';
```

Die Verknüpfung dieser beiden Zeichenketten ergibt den einzelnen Wert Hi Ivy, der in der Variablen \$greeting gespeichert wird.

Im weiteren Verlauf dieses Kapitels lernen Sie die auf der rechten Seite aufgeführten Operatoren kennen.

ARITHMETISCHE OPERATOREN

Seite 50 bis 51

Mit arithmetischen Operatoren können Sie Zahlen verarbeiten und beispielsweise Additionen, Subtraktionen, Multiplikationen und Divisionen durchführen.

Wenn zum Beispiel jemand 3 Päckchen Süßigkeiten kauft und jedes Päckchen 5\$ kostet, können Sie mit einem Multiplikationsoperator den Gesamtpreis für diese drei Süßigkeitenpakete berechnen.

VERGLEICHSOPERATOREN

Seite 54 bis 55 und 58

Wie der Name schon sagt, vergleichen Vergleichsoperatoren zwei Werte und geben einen booleschen Wert (entweder true oder false) zurück.

Im Falle der beiden Zahlen 3 und 5 könnten Sie mit einem Vergleich beispielsweise feststellen, ob:

- 3 größer als 5 ist (false)
- 3 gleich 5 ist (false)
- 3 kleiner als 5 ist (true)

Auch Zeichenketten können Sie vergleichen, um festzustellen, ob ein Wert größer oder kleiner als ein anderer ist:

- 'Apple' ist größer als 'Banana' (false)
- 'A' ist gleich 'B' (false)
- 'A' ist kleiner als 'B' (true)

STRING-OPERATOREN

Seite 52 bis 53

Mit String-Operatoren können Sie Texte verarbeiten. Es gibt zwei String-Operatoren, mit denen Sie verschiedene Textstücke zu einer einzigen Zeichenkette zusammenfügen können.

Wenn Sie zum Beispiel den Vornamen eines Mitglieds in einer Variablen und den Nachnamen in einer zweiten Variablen gespeichert haben, können Sie diese beiden Variablen miteinander kombinieren, um den vollständigen Namen zu erhalten.

LOGISCHE OPERATOREN

Seite 56 bis 57 und 59

Die drei logischen Operatoren and, or und not beziehen sich immer auf zwei Eingabewerte, entweder true oder false. Um ihre Funktionsweise zu verstehen, betrachten Sie die beiden folgenden Fragen; beide können mit wahr oder falsch beantwortet werden:

Ist es heiß? Ist es sonnig?

- Der and-Operator prüft, ob es zugleich heiß **und** sonnig ist.
- Der or-Operator prüft, ob es entweder heiß **oder** sonnig ist.
- Mit dem not-Operator können Sie prüfen, ob jeweils nur die Antwort auf eine dieser Fragen nicht zutrifft.
- Zum Beispiel: Ist es nicht sonnig?

Als Ergebnis erhalten Sie jeweils entweder den Wert true oder false.

ARITHMETISCHE OPERATOREN

In PHP können Sie die folgenden mathematischen Operatoren mit Zahlen und Variablen, in denen Zahlen gespeichert sind, verwenden.

NAME	OPERATOR	ZWECK	BEISPIEL	ERGEBNIS
Addition	+	Einen Wert zu einem anderen hinzuzählen	10 + 5	15
Subtraktion	-	Einen Wert von einem anderen abziehen	10 - 5	5
Multiplikation	*	Zwei Werte miteinander multiplizieren	10 * 5	50
Division	/	Einen Wert durch einen anderen teilen	10 / 5	2
Modulo	%	Einen Wert durch einen anderen teilen und den Rest ausgeben	10 % 3	1
Potenzierung	**	Einen Wert mit einem anderen potenzieren	10 ** 5	100000
Erhöhung	++	Den Zahlenwert um eins erhöhen	\$i = 10; \$i++;	11
Verringerung	--	Den Zahlenwert um eins absenken	\$i = 10; \$i--;	9

AUSFÜHRUNGSREIHENFOLGE

Sie können innerhalb eines Ausdrucks mehrere arithmetische Operationen durchführen, aber dabei müssen Sie die Reihenfolge beachten, in der das Ergebnis berechnet wird: Multiplikationen und Divisionen werden vor Additionen und Subtraktionen ausgeführt.

Dies kann Auswirkungen auf das von Ihnen erwartete Ergebnis haben. Die Zahlen hier werden beispielweise von links nach rechts berechnet. Das Ergebnis ist 16:

```
$total = 2 + 4 + 10;
```

Im folgenden Beispiel ist das Ergebnis jedoch 42 (nicht 60):

```
$total = 2 + 4 * 10;
```

Mithilfe von Klammern können Sie bestimmen, welche Operation zuerst durchgeführt werden soll. Die folgende Berechnung liefert also ein Ergebnis von 60:

```
$total = (2 + 4) * 10;
```

Die Klammern zeigen an, dass zunächst 2 und 4 zusammengezählt werden, bevor das Ergebnis mit 10 multipliziert wird.

ARITHMETISCHE OPERATOREN VERWENDEN

PHP

section_a/c01/arithmetic-operators.php

```
<?php  
① $items      = 3;  
② $cost       = 5;  
③ $subtotal   = $cost * $items;  
④ $tax        = ($subtotal / 100) * 20;  
⑤ $total      = $subtotal + $tax;  
?  
<!DOCTYPE html>  
  <html>  
    <head> ... </head>  
    <body>  
      <h1>The Candy Store</h1>  
      <h2>Shopping Cart</h2>  
      <p>Items: <?= $items ?></p>  
      <p>Cost per pack: $<?= $cost ?></p>  
      <p>Subtotal: $<?= $subtotal ?></p>  
      <p>Tax: $<?= $tax ?></p>  
      <p>Total: $<?= $total ?></p>  
    </body>  
  </html>
```

ERGEBNIS



Dieses Beispiel zeigt, wie mathematische Operatoren in Verbindung mit Zahlen eingesetzt werden, um den Gesamtpreis einer Bestellung zu berechnen. Zunächst benötigen Sie zwei Variablen zur Speicherung der:

1. Gesamtzahl der bestellten Artikel (`$items`)
 2. Kosten für eine einzelne Packung Süßigkeiten (`$cost`)
- Anschließend erfolgen die Berechnungen, und die Ergebnisse werden in Variablen gespeichert. Erst danach wird der HTML-Code erstellt. Dies hilft, den PHP-Code vom HTML-Inhalt zu trennen.
3. Zur Berechnung der Auftragskosten wird die Anzahl der Artikel mit dem Preis für eine Packung Süßigkeiten multipliziert.
 4. Nachfolgend muss die Steuer in Höhe von 20 % hinzugerechnet werden. Dazu wird die Zwischensumme durch 100 geteilt. (Dies geschieht in Klammern, um sicherzustellen, dass sie zuerst berechnet wird.) Anschließend wird das Ergebnis mit 20 multipliziert.
 5. Schließlich wird die Steuer zur Zwischensumme hinzugefügt, um den Gesamtbetrag zu ermitteln.
 6. Die in Variablen abgelegten Ergebnisse werden dann auf der HTML-Seite ausgegeben.

Probieren Sie es: Verändern Sie die Stückpreise in Schritt 1 und die Menge in Schritt 2.

ZEICHENKETTEN-OPERATOREN

Möglicherweise müssen Sie zwei oder mehrere Zeichenketten zusammenfügen, um daraus einen einzigen Wert zu erzeugen.

Eine solche Operation wird als Verknüpfung oder Verkettung von Strings bezeichnet.



VERKKNÜPFUNGSOOPERATOR

Der Verknüpfungsoperator ist ein Punktsymbol. Er verknüpft den Wert in einer Zeichenkette mit dem Wert in einer weiteren. Im folgenden Beispiel würde der Variablen \$name die Zeichenfolge 'Ivy Stone' zugewiesen:

```
$forename = 'Ivy';
$surname = 'Stone';
$name     = $forename . ' ' . $surname;
```

Beachten Sie, dass zwischen die Variablen \$forename und \$surname noch ein separates Leerzeichen eingefügt wird; ohne dieses Leerzeichen würde die Variable \$name den Wert IvyStone enthalten. Sie können beliebig viele Zeichenketten in einer Anweisung verknüpfen, sofern Sie zwischen den einzelnen Zeichenketten konsequent den Verknüpfungsoperator anwenden.

Sie können in Variablen gespeicherte Zeichenketten auch ganz ohne Verknüpfungsoperator zusammenfügen. Wenn ein Wert mit doppelten (statt einfachen) Anführungszeichen zugewiesen wird, ersetzt der PHP-Interpreter die Variablennamen in doppelten Anführungszeichen durch die darin enthaltenen Werte. Im folgenden Fall würde \$name daher den Wert Ivy Stone enthalten:

```
$name = "$forename $surname";
```

VERKNÜPFENDER ZUWEISUNGSOOPERATOR

Wenn Sie einen Text an eine bereits vorhandene Variable anhängen möchten, können Sie den zuweisenden Verknüpfungsoperator verwenden. Diesen können Sie sich als Kurzform zur Erzeugung eines aktualisierten Strings vorstellen:

```
$greeting = 'Hello ';
$greeting .= 'Ivy';
```

Hier wird die Zeichenkette 'Hello ' in der Variablen \$greeting gespeichert. In der nächsten Zeile fügt der verknüpfende Zuweisungsoperator die Zeichenkette 'Ivy' hinten an den bestehenden Wert der Variablen \$greeting an.

Jetzt lautet der Wert der Variablen \$greeting 'Hello Ivy'. Wie Sie sehen, spart dies gegenüber dem links gezeigten Beispiel eine Zeile Code ein.

ZEICHENKETTEN ZUSAMMENFÜGEN

PHP

```
<?php  
① $prefix = 'Thank you';  
② $name = 'Ivy';  
③ $message = $prefix . ', ' . $name;  
?>  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>String Operator</title>  
    <link rel="stylesheet" href="css/styles.css">  
  </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2><?= $name ?>'s Order</h2>  
    <p><?= $message ?></p>  
  </body>  
</html>
```

ERGEBNIS



Dieses Beispiel zeigt eine personalisierte Nachricht an.

1. Zunächst wird die Variable `$prefix` erstellt, um den Anfang der Nachricht zu speichern. Sie enthält die Worte 'Thank you'.
2. Eine zweite neu erstellte Variable nimmt den Namen des Besuchers oder der Besucherin auf. Die Variable heißt `$name` und die Besucherin heißt Ivy.
3. Die persönliche Nachricht wird durch Verknüpfung (oder Verkettung) dreier Werte erzeugt und der neue Wert in der Variablen `$message` gespeichert:
 - Zunächst wird der in `$prefix` gespeicherte Wert zu `$message` hinzugefügt
 - Dann werden ein Komma und ein Leerzeichen angehängt
 - Schließlich wird der in `$name` gespeicherte Wert angefügt

Probieren Sie es: Ändern Sie in Schritt 2 den in `$name` gespeicherten Wert in Ihren Namen um.

Probieren Sie es: Weisen Sie der Variablen `$message` in Schritt 3 ihren Wert mithilfe doppelter Anführungszeichen (und ohne Verknüpfungsoperator) zu:

```
$message = "$prefix  
$name";
```

VERGLEICHSOPERATOREN VERWENDEN

Mithilfe von Vergleichsoperatoren können Sie zwei oder mehr Werte miteinander vergleichen. Das Ergebnis ist ein boolescher Wert, der entweder wahr oder falsch (`true` oder `false`) ist.

==

IST GLEICH

Dieser Operator vergleicht zwei Werte, um festzustellen, ob sie gleich sind.

`'Hello' == 'Hello'` ergibt `true`,
da es sich um denselben String handelt.
`'Hello' == 'Goodbye'` ergibt `false`,
da die beiden Strings unterschiedlich sind.

!= ODER <>

IST NICHT GLEICH

Diese Operatoren vergleichen zwei Werte, um festzustellen, ob sie ungleich sind.

`'Hello' != 'Hello'` ergibt `false`
da es sich um denselben String handelt.
`'Hello' != 'Goodbye'` ergibt `true`
da die beiden Strings unterschiedlich sind.

Mit den oben genannten Operatoren kann der PHP-Interpreter feststellen, ob die beiden Werte einander gleich sind oder nicht. Die nachfolgenden Operatoren sind strikter, da sie sowohl den Wert als auch den Datentyp überprüfen.

Die oben genannten Operatoren würden 3 (eine ganze Zahl) und 3.0 (eine Fließkommazahl) als gleich betrachten. Für die nachfolgenden Operatoren gilt dies nicht. (Auf den Seiten 60 und 61 sehen Sie, wie 0 auch als boolescher Wert `false` und 1 als `true` interpretiert werden kann).

====

IST IDENTISCH MIT

Dieser Operator vergleicht zwei Werte, um festzustellen, ob sowohl deren Wert als auch deren Datentyp übereinstimmen.

`'3' === 3` ergibt `false`,
weil es sich nicht um denselben Datentyp handelt.
`'3' === '3'` ergibt `true`
weil sowohl Datentyp als auch Wert übereinstimmen.

!= **==**

IST NICHT IDENTISCH MIT

Dieser Operator vergleicht zwei Werte, um zu prüfen, ob sie im Wert und/oder Datentyp voneinander abweichen.

`3.0 !== 3` ergibt `true`
weil es sich nicht um denselben Datentyp handelt.
`3.0 !== 3.0` ergibt `false`
weil sowohl Datentyp als auch Wert übereinstimmen.

Wenn Sie einen booleschen Wert mittels echo auf der Seite ausgeben, wird bei true eine 1 angezeigt und bei false gar nichts.



KLEINER ALS und GRÖSSER ALS

< prüft, ob der Wert auf der linken Seite kleiner ist als der Wert auf der rechten Seite.

4 < 3 ergibt false 3 < 4 ergibt true

> prüft, ob der Wert auf der linken Seite größer ist als der Wert auf der rechten Seite.

z > a ergibt true a > z ergibt false



KLEINER ODER GLEICH und GRÖSSER ODER GLEICH

<= prüft, ob der Wert auf der linken Seite kleiner oder gleich dem Wert auf der rechten Seite ist.

4 <= 3 ergibt false 3 <= 4 ergibt true

>= prüft, ob der Wert auf der linken Seite größer oder gleich dem Wert auf der rechten Seite ist.

z >= a ergibt true z >= z ergibt true



RAUMSCHIFF-OPERATOR

Der Raumschiff-Operator vergleicht die links und rechts von ihm stehenden Werte und liefert:

0 für zwei gleiche Werte

1 für einen größeren Wert auf der linken Seite

-1 für einen größeren Wert auf der rechten Seite

Dieser Operator wurde erst mit PHP 7 eingeführt (und funktioniert mit früheren PHP-Versionen nicht).

1 <=> 1 ergibt: 0

2 <=> 1 ergibt: 1

2 <=> 3 ergibt: -1

LOGISCHE OPERATOREN

Vergleichsoperatoren liefern einen einzelnen Wert, der entweder `true` oder `false` ist. Mehrere Vergleichsoperatoren können zusammen mit logischen Operatoren eingesetzt werden, um die Ergebnisse mehrerer Ausdrücke zu vergleichen.

In dieser einen Codezeile stehen drei Ausdrücke, von denen jeder einen einzigen Wert (entweder `true` oder `false`) liefert.

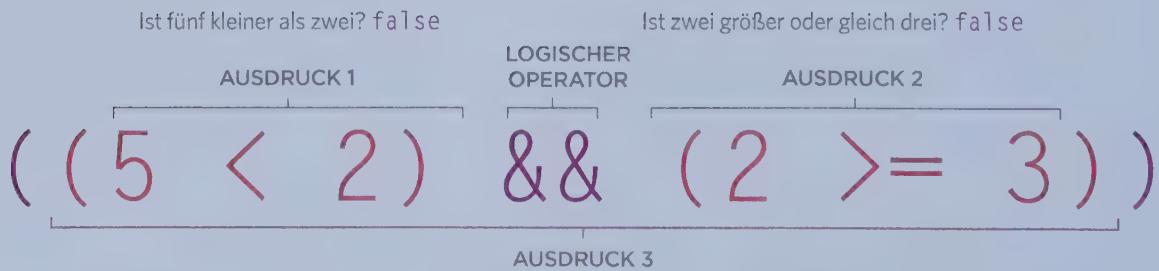
Ausdruck 1 (links) und Ausdruck 2 (rechts) enthalten beide einen Vergleichsoperator, und beide Ausdrücke liefern den Wert `false`.

Ausdruck 3 nutzt einen logischen Operator (und keinen Vergleichsoperator).

Der logische Und-Operator (`&&`) prüft, ob die Ausdrücke auf beiden Seiten den Wert `true` liefern. In diesem Fall trifft das nicht zu, sodass der gesamte Ausdruck den Wert `false` ergibt.

Die Ausdrücke 1 und 2 werden vor Ausdruck 3 ausgewertet.

Jeder Ausdruck steht für sich in Klammern. Auf diese Weise wird besser deutlich, dass der Code in jedem einzelnen Klammerpaar einen einzigen Wert ergeben sollte. Es funktioniert auch ohne die Klammern, ist dann aber viel schlechter lesbar.



Sind die Ausdrücke 1 und 2 jeweils wahr (`true`)? `false`

&&

LOGISCHES UND

Dieser Operator überprüft mehrere Bedingungen.

((2 < 5) && (3 >= 2))

Dies ergibt den Wert true.

Wenn beide Ausdrücke den Wert true haben, ergibt der Ausdruck true.

Wenn einer der Ausdrücke den Wert false hat, liefert der Ausdruck ebenfalls den Wert false.

true && true ergibt true

true && false ergibt
false

false && true ergibt
false

false && false ergibt
false

Statt der beiden kaufmännischen Und-Zeichen können Sie auch das Wort and verwenden.



!

LOGISCHES NICHT

Dieser Operator negiert einen einzelnen booleschen Wert.

!(2 < 1)

Dies ergibt den Wert true.

Das ! negiert einen Ausdruck. Wenn er false ist (ohne das ! davor), ergibt sich dadurch true.

Wenn die Aussage true ist, ergibt sich false.

!true ergibt false

!false ergibt true

Sie können statt des Ausrufezeichen nicht das Wort not verwenden.

VERKÜRZTE AUSWERTUNG

Logische Ausdrücke werden von links nach rechts ausgewertet. Sobald der erste Ausdruck ausgewertet wurde und der PHP-Interpreter den logischen Operator kennt, kann er möglicherweise auf die Auswertung der zweiten Bedingung verzichten. Warum, sehen Sie in den Beispielen rechts.

((5 < 2) && (2 >= 2))



Ein falscher Wert (false) wurde gefunden.

Es hat keinen Sinn, die zweite Bedingung zu überprüfen, da die beiden Bedingungen zusammen nicht mehr den Wert true ergeben können.

((2 < 5) || (2 >= 2))



Ein wahrer Wert (true) wurde gefunden.

Es hat keinen Sinn, die zweite Bedingung zu überprüfen, da mindestens einer der Werte true lautet.

VERGLEICHSOPERATOREN VERWENDEN

1. Es werden drei Variablen angelegt:

- Die erste enthält die Art der gewünschten Süßigkeiten.
- Die zweite zeigt, dass 5 Päckchen auf Lager sind.
- Die dritte zeigt, dass 8 Päckchen gewünscht sind.

2. Ein Vergleichsoperator überprüft, ob die gewünschte Menge kleiner oder gleich der verfügbaren Menge ist. Das Ergebnis wird in der Variablen \$can_buy gespeichert.

3. Sie werden kaum jemals einen booleschen Wert in eine Seite schreiben (so wie es hier gezeigt wird). Viel wahrscheinlicher werden Sie den Wert in einer bedingten Logik einsetzen, wie Sie sie im nächsten Kapitel kennenlernen. Aber es ist wichtig zu sehen, was Sie beim Versuch, einen solchen booleschen Wert auszuschreiben, erhalten:

- für true zeigt die Seite 1 an
- für false zeigt die Seite nichts an

Probieren Sie es: Vertauschen Sie in Schritt 1 die Werte in \$stock und \$wanted. Dadurch ändert sich der Wert in \$can_buy.

Auf Seite 75 erfahren Sie, wie Sie verschiedene Meldungen anzeigen können, wenn ein Ausdruck mit Vergleichsoperator true oder false zurückliefert.

section_a/c01/comparison-operators.php

PHP

```
<?php
    $item      = 'Chocolate';
    $stock     = 5;
    $wanted    = 8;
    ② $can_buy = ($wanted <= $stock);
?
<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Shopping Cart</h2>
        <p>Item: <?= $item ?></p>
        <p>Stock: <?= $stock ?></p>
        <p>Wanted: <?= $wanted ?></p>
        <p>Can buy: <?= $can_buy ?></p>
    </body>
</html>
```

ERGEBNIS



LOGISCHE OPERATOREN VERWENDEN

PHP

section_a/c01/logical-operators.php

```
<?php
$item    = 'Chocolate';
$stock   = 5;
① $wanted  = 3;
② $deliver = true;
③ $can_buy = (($wanted <= $stock) && ($deliver == true));
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item: <?= $item ?></p>
    <p>Stock: <?= $stock ?></p>
    <p>Ordered: <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>
  </body>
</html>
```

ERGEBNIS



Dieses Beispiel baut auf dem links gezeigten Beispiel auf.

1. Die Kundin möchte hier nur 3 Päckchen Süßwaren kaufen.
2. Die Variable \$deliver wird angelegt; sie speichert einen booleschen Wert, der angibt, ob eine Lieferung möglich ist oder nicht.

3. Dieser Ausdruck verwendet zwei Vergleichsoperatoren:

- Die erste überprüft, ob genügend Artikel auf Lager sind
- Die zweite überprüft, ob der Artikel geliefert werden kann

Mit dem logischen Operator && wird geprüft, ob beide Bedingungen den Wert true ergeben. Wenn dies der Fall ist, ist der Wert von \$can_buy true, und auf der Seite wird die Zahl 1 ausgegeben.

Sind nicht beide Bedingungen erfüllt, enthält \$can_buy den Wert false und es wird gar nichts angezeigt.

Probieren Sie es: Vertauschen Sie in Schritt 1 die Werte in \$stock und \$wanted. Dadurch ändert sich der Wert in \$can_buy.

Auf Seite 75 erfahren Sie, wie Sie verschiedene Meldungen anzeigen können, wenn ein Ausdrucks true oder false zurückliefert.

TYPE JUGGLING: AUTOMATISCHE TYPUMWANDLUNG

Der PHP-Interpreter kann einen Wert von einem Datentyp in einen anderen umwandeln. Dies wird auch als »Type Juggling« bezeichnet und kann zu unerwarteten Ergebnissen führen.

PHP ist als Sprache mit **schwacher Typisierung** bekannt, weil Sie bei der Variablen Deklaration keinen Datentyp für den zu speichernden Wert angeben müssen. Im Folgenden enthält die Variable \$title zunächst eine Zeichenkette und dann eine ganze Zahl:

```
$title = 'Ten'; // String  
$title = 10; // Integer
```

Im Unterschied dazu müssen Programmierer in Programmiersprachen mit **strengerer Typisierung** (wie z.B. C++ oder C#) bei der Deklaration jeder Variablen den Datentyp angeben, den sie enthalten soll.

Wenn der PHP-Interpreter auf einen Wert stößt, der nicht den erwarteten Datentyp enthält, kann er versuchen, den Wert in den erwarteten Datentyp zu konvertieren. Dieser Prozess wird auch als **Type Juggling** oder Type Casting bezeichnet.

Die automatische Typumwandlung kann zu Problemen führen, da der PHP-Interpreter mitunter überraschende Ergebnisse oder Fehler erzeugt. Der unten stehende Additionsoperator addiert zum Beispiel zwei Werte zusammen. Die Zahl 1 ist eine Ganzzahl, die 2 ist allerdings ein String, weil sie in Anführungszeichen steht.

```
$total = 1 + '2';
```

In diesem Fall versucht der PHP-Interpreter automatisch, die Zeichenkette in eine Zahl umzuwandeln, damit er die Berechnung durchführen kann. Im Ergebnis enthält die Variable \$total daher die Zahl 3.

Auf der rechten Seite sehen Sie, nach welchen Regeln ein Wert von einem Datentyp in einen anderen umgewandelt wird. Beispiele für das Type Juggling finden Sie unter: <http://notes.re/php/type-juggling>.

Wenn der Datentyp eines Werts in einen anderen Datentyp umgewandelt wird, sprechen Programmierer davon, dass der Datentyp des Werts von einem Typ in einen anderen **gecastet** wird. Das Type Juggling wird als implizites Casting oder **implizite Typumwandlung** bezeichnet, da es vom PHP-Interpreter selbstständig durchgeführt wird.

Wenn Sie den Datentyp eines Werts explizit mittels Code verändern, wird dies als explizites Casting oder **explizite Typumwandlung** bezeichnet, da der PHP-Interpreter explizit angewiesen wurde, den Datentyp zu ändern.

ZAHLEN

Wenn der PHP-Interpreter zwei Zahlen erwartet, kann er eine mathematische Operation mit diesen durchführen.

Nachfolgend sehen Sie, was passiert, wenn:

- eine Zeichenkette zu einer Zahl addiert wird
- ein boolescher Wert zu einer Zahl addiert wird

ZAHL + STRING	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
1 + '1'	1 + 1	2 (int)	String enthält eine gültige Ganzzahl. Sie wird als Ganzzahl behandelt.
1 + '1.2'	1 + 1.2	2.2 (float)	String enthält einen Fließkommawert. Er wird als Fließkommawert behandelt.
1 + '1.2e+3'	1 + 1200	1201 (float)	String enthält einen Fließkommawert mit einem e (Zehnerpotenz). Er wird als Fließkommawert behandelt.
1 + '5star'	1 + 5	6 (int)	String enthält eine Ganzzahl, gefolgt von weiteren Zeichen. Die Zahl wird als Ganzzahl behandelt. Nachfolgende Zeichen werden ignoriert.
1 + '3.5star1'	1 + 3.5	4.5 (float)	String enthält einen Fließkommawert, gefolgt von weiteren Zeichen. Die Zahl wird als Fließkommawert behandelt. Nachfolgende Zeichen werden ignoriert.
1 + 'star9'	1 + 0	1 (int)	String beginnt mit etwas anderem als einer Ganzzahl oder einem Fließkommawert. Er wird wie die Zahl 0 behandelt.

ZAHL + BOOLESCHE WERT	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
1 + true	1 + 1	2 (int)	Boolescher Wert true wird als Ganzzahl 1 behandelt.
1 + false	1 + 0	1 (int)	Boolescher Wert false wird als Ganzzahl 0 behandelt.

ZEICHENKETTEN

Wenn der PHP-Interpreter versucht, zwei Zeichenketten zu verknüpfen, folgt er diesen Regeln.

Nachfolgend sehen Sie, was passiert, wenn PHP:

- eine Zeichenkette mit einer Zahl verknüpft
- eine Zeichenkette mit einem booleschen Wert verknüpft

STRING . ZAHL	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
'Hi ' . 1	'Hi ' . '1'	Hi 1 (string)	Ganzzahl wird als String behandelt.
'Hi ' . 1.23	'Hi ' . '1.23'	Hi 1.23 (string)	Fließkommazahl wird als String behandelt.

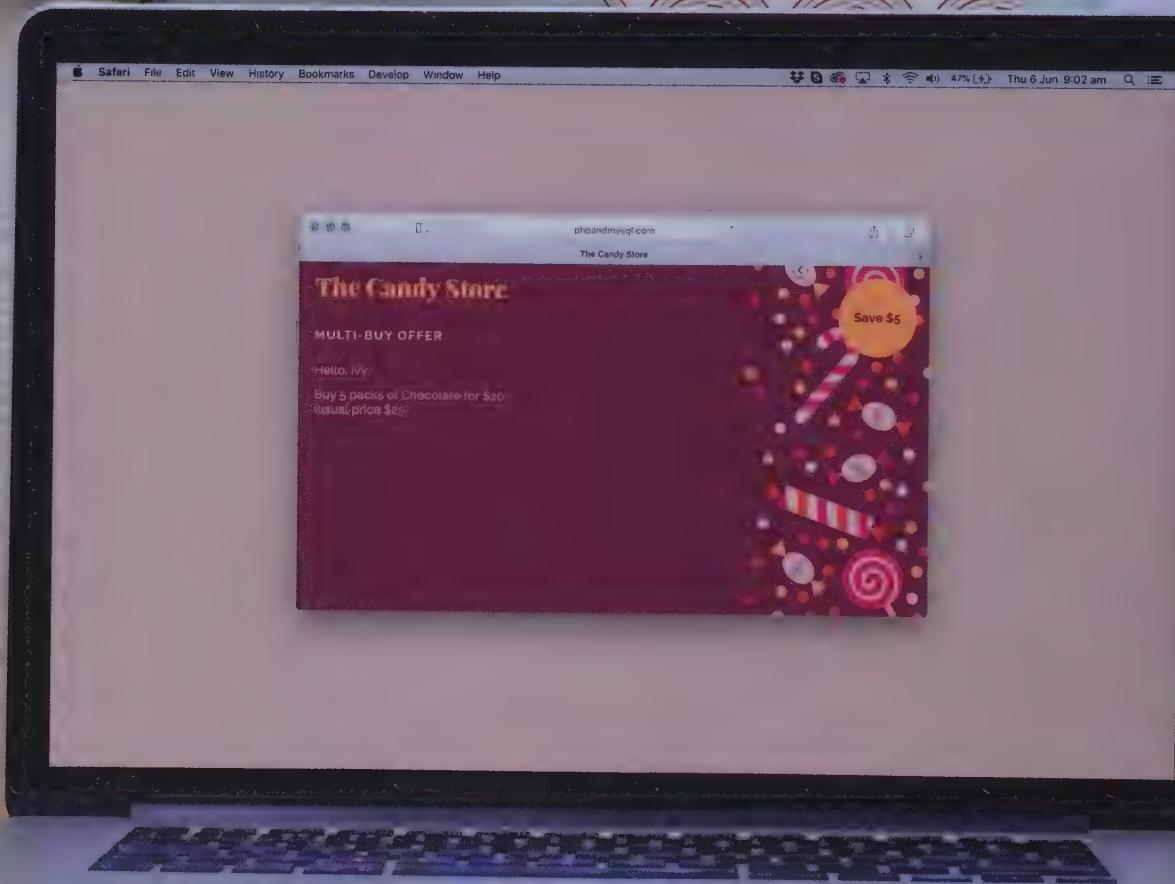
STRING . BOOLESCHE WERT	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
'Hi ' . true	'Hi ' . '1'	Hi 1 (string)	Boolescher Wert true wird als Integer-Wert 1 behandelt.
'Hi ' . false	'Hi ' . ''	Hi (string)	Boolescher Wert false wird als leerer String behandelt.

WERT	DATENTYP	BEHANDELT ALS
false	Boolean	false
0	Integer	false
0.0	Float	false
'0'	String mit Wert 0	false
''	Leerer String	false
array[]	Leeres Array	false
null	Null	false

BOOLESCHE WERTE

Wenn der PHP-Interpreter einen booleschen Wert erwartet, werden alle in der Tabelle links aufgeführten Werte als false behandelt.

Sämtliche anderen Werte werden als true behandelt (jeglicher Text, jede Zahl ungleich 0 oder der boolesche Wert true).



EINE EINFACHE PHP-SEITE

Dieses Beispiel fasst mehrere der Techniken zusammen, die Sie in diesem Kapitel kennengelernt haben.

Die PHP-Datei erzeugt eine HTML-Seite, die über einen Rabatt beim Kauf mehrerer Päckchen Süßigkeiten informiert.

Sie werden sehen, wie Sie:

- Informationen in Variablen und Arrays speichern.
- den Verknüpfungsoperator verwenden, um Texte in Variablen zu verbinden und so eine personalisierte Begrüßung zu erstellen.
- arithmetische Operatoren einsetzen, um die auf der Seite angezeigten Preise zu berechnen.
- die vom PHP-Interpreter neu erstellten Werte in den HTML-Bereich der Seite schreiben.

Zudem gibt die Seite automatisch die neuen Produkte und Preise wieder, wenn die in den Variablen gespeicherten Werte aktualisiert werden.

DATEN VERARBEITEN UND ANZEIGEN

Wenn Sie anfangen, PHP-Dateien zu schreiben, enthalten diese oft eine Mischung aus HTML- und PHP-Code. Es empfiehlt sich dabei, diesen Code so weit wie möglich zu trennen:

- Fangen Sie mit PHP an, um die Werte zu erzeugen, die auf der HTML-Seite angezeigt werden sollen, und speichern Sie diese in Variablen. (Rechts entspricht dies dem Bereich oberhalb der gestrichelten Linie.)
- Der untere Teil der Seite kann sich dann auf den HTML-Inhalt konzentrieren. PHP-Code sollte hier lediglich zur Anzeige der in den Variablen abgelegten Werte verwendet werden. (Rechts entspricht dies dem Bereich unterhalb der gestrichelten Linie.)

Sehen wir uns den PHP-Code am Anfang der Seite an:

1. Dieses Beispiel beginnt mit der Deklaration einer Variablen zur Speicherung des Benutzernamens. Sie heißt \$username, weil ein Variablenname immer mit einem Dollarzeichen beginnen sollte, gefolgt von einem aussagekräftigen Namen, der die Art der darin enthaltenen Daten beschreibt.
2. Hier wird die Variable \$greeting deklariert, in der eine Begrüßungsfloskel hinterlegt wird. Diese wird mithilfe des String-Operators aus der Zeichenkette Hello und dem Benutzernamen zusammengesetzt.
3. Die Variable \$offer wird angelegt, um die Details eines im Sonderangebot befindlichen Artikels zu speichern. Als Wert enthält sie ein Array mit vier Elementen:

- der angebotene Artikel
- die erforderliche Abnahmemenge
- der normale Packungspreis (ohne Rabatt)
- der ermäßigte Packungspreis

Das erste Element, das den angebotenen Artikel beschreibt, ist vom Datentyp String. Die anderen Werte sind ganzzahlige Integer.

4. Die Variable \$usual_price wird angelegt. Ihr Wert ist der Artikelpreis ohne Rabatt. Er wird durch die Multiplikation zweier im Array gespeicherter Werte berechnet: der Menge und dem Preis.
5. Die Variable \$offer_price wird angelegt. Ihr Wert ist der rabattierte Preis der Artikel. Dieser wird durch Multiplikation der Menge und des ermäßigten Packungspreises (beides im Array gespeichert) berechnet.
6. Die Variable \$saving wird angelegt, um die Gesamtersparnis zu speichern. Diese berechnet sich durch Subtraktion des in der Variablen \$offer_price (erstellt in Schritt 5) gespeicherten Werts von dem in \$usual_price (erstellt in Schritt 4) gespeicherten Wert.
In der zweiten Hälfte der Seite (unterhalb der gestrichelten Linie) wird das HTML generiert, das an den Browser zurückgesendet wird. Sie beginnt mit der Deklaration des HTML-DOCTYPE. PHP dient hier nur noch dazu, Werte auszugeben, die in den vorangegangenen Schritten in Variablen gespeichert wurden:
7. Die Begrüßung, also das Wort Hello, gefolgt vom Benutzernamen, wird mit der Kurzschrifweisweise des echo-Befehls auf der Seite ausgegeben.
8. Die Gesamtersparnis, die in der (in Schritt 6 erstellten) Variablen \$saving gespeichert ist, wird in einem gelben Kreis angezeigt. Mithilfe von CSS wird dieser Kreis rechts oben im Browserfenster platziert.
9. Ein neuer Absatz erläutert die Einzelheiten des Angebots.
Er zeigt die Menge und den Namen der Süßigkeit, die der Besucher kaufen muss.
10. Darauf folgen der in \$offer_price gespeicherte ermäßigte Preis und der in \$usual_price abgelegte Normalpreis.

```

<?php
① $username = 'Ivy';                                // Variable für Benutzernamen

② $greeting = 'Hello, ' . $username . '.';          // Begrüßung lautet 'Hello, ' + Benutzername

③ $offer = [                                         // Array für Angebotsdaten anlegen
    'item'    => 'Chocolate',                         // angebotener Artikel
    'qty'     => 5,                                    // Mindestmenge
    'price'   => 5,                                    // Normalpreis pro Stück
    'discount' => 4,                                  // Angebotspreis pro Stück
];
                                         // Angebotspreis gesamt
④ $usual_price = $offer['qty'] * $offer['price'];      // Normalpreis gesamt
⑤ $offer_price = $offer['qty'] * $offer['discount'];    // Angebotspreis gesamt
⑥ $saving      = $usual_price - $offer_price;           // Ersparnis gesamt
?>
-----<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>

    <h2>Multi-buy Offer</h2>

    ⑦ <p><?= $greeting ?></p>

    ⑧ <p class="sticker">Save $<?= $saving ?></p>

    ⑨ <p>Buy <?= $offer['qty'] ?> packs of <?= $offer['item'] ?>
    ⑩       for $<?= $offer_price ?><br>(usual price $<?= $usual_price ?>)</p>
  </body>
</html>

```

Probieren Sie es: Ändern Sie in Schritt 1 den Benutzernamen in Ihren Namen um.

Ändern Sie in Schritt 2 die Begrüßung in Hi (statt Hello).

Ändern Sie in Schritt 3 die Päckchenanzahl im qty-Schlüssel des Arrays \$offer auf 3.

Ändern Sie in Schritt 3 den Süßwarenpreis auf 6.

ZUSAMMENFASSUNG

VARIABLEN, AUSDRÜCKE & OPERATOREN

- Variablen speichern Daten, die sich bei jeder Ausführung eines Skripts verändern können.
- Skalare Datentypen speichern Text, ganze Zahlen, Fließkommazahlen und die booleschen Werte true oder false.
- Ein Array ist ein zusammengesetzter Datentyp zum Speichern einer Reihe von zusammengehörigen Werten.
- Die einzelnen Einträge in einem Array werden als Elemente bezeichnet. Elemente in assoziativen Arrays verfügen über einen Schlüssel und einen Wert. Elemente in indizierten Arrays haben eine Indexnummer und einen Wert.
- String-Operatoren verknüpfen (verketten) Texte in Zeichenketten.
- Mathematische Operatoren sind für mathematische Berechnungen mit Zahlen zuständig.
- Vergleichsoperatoren vergleichen zwei Werte, um festzustellen, ob sie beide gleich oder einer größer oder kleiner als der andere ist.
- Logische Operatoren verknüpfen die Ergebnisse mehrerer Ausdrücke mit UND, ODER und NICHT.

2

KONTROLL- STRUKTUREN

In diesem Kapitel erfahren Sie, wie Sie dem PHP-Interpreter mitteilen können, ob er einen bestimmten Code-Block ausführen soll oder nicht, wann er eine Reihe von Anweisungen wiederholen und wann Code aus einer anderen Datei eingebunden werden soll.

Es gibt drei Möglichkeiten zu steuern, wann der PHP-Interpreter Anweisungen in einer PHP-Datei ausführt:

- **Reihenfolge:** Der PHP-Interpreter führt die Anweisungen in der Reihenfolge aus, in der sie formuliert sind. Zeile 1, Zeile 2, Zeile 3 und so weiter, bis die letzte Zeile erreicht ist. Alle bisherigen Beispiele in diesem Buch arbeiten den Code in dieser Reihenfolge ab.
- **Auswahl:** Der PHP-Interpreter überprüft anhand einer Bedingung, ob ein bestimmter Code-Bereich ausgeführt werden soll oder nicht. Die Bedingung könnte zum Beispiel lauten: »Ist ein Anwender eingeloggt?« Wenn die Antwort »Nein« lautet, könnte ein Link zur Anmeldeseite angezeigt werden. Lautet die Antwort dagegen »Ja«, könnte ein Link zur Profilseite erscheinen. Programmierer sprechen hier von bedingten Anweisungen, da anhand einer Bedingung ausgewählt wird, welche Gruppe von Anweisungen ausgeführt werden soll.
- **Wiederholung/Iteration:** Der PHP-Interpreter kann denselben Code mehrfach durchlaufen. Wenn ein Array beispielsweise eine Einkaufsliste enthält, können die gleichen Anweisungen für jedes einzelne Listenelement ausgeführt werden (unabhängig davon, ob es sich dabei um 1 oder 100 Einträge handelt). Zur wiederholten Ausführung von Anweisungen werden Schleifen eingesetzt.

Wenn Sie die Ausführungsreihenfolge der Anweisungen ändern, verändern Sie damit den Programmablauf oder den Kontrollfluss.

In diesem Kapitel lernen Sie auch den Einsatz von Include-Dateien, in denen Code gespeichert ist, der von verschiedenen Seiten abgerufen wird. Mit dieser Methode können Sie eine Datei in mehrere Seiten einbinden, ohne denselben Code in jeder Datei erneut zu wiederholen.



BEDINGTE ANWEISUNGEN

Bei **bedingten Anweisungen** wird eine **Bedingung** überprüft, um festzustellen, ob ein Programmbaustein ausgeführt werden soll oder nicht. Man könnte auch sagen: »Wenn diese Situation zutrifft, führe Aktion 1 aus«. (Und bei Bedarf: »Wenn nicht, führe Aktion 2 aus«).

Einige Aktionen werden nur dann ausgeführt, wenn eine gewisse Voraussetzung erfüllt ist. Nehmen wir eine Website, auf der sich Benutzer anmelden können. Wenn der Benutzer:

- *eingeloggt ist*, wird ein Link zu seiner Profilseite angezeigt.
- *nicht eingeloggt ist*, erscheint ein Link zur Anmeldeseite.

Hier lautet die Bedingung also: »Ist der Benutzer eingeloggt?« und sie dient der Entscheidung, welcher Link angezeigt werden soll.

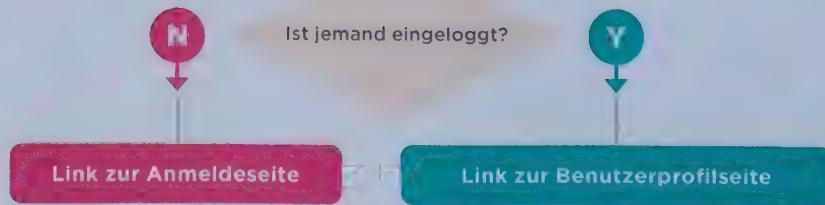
Bedingungen sind Ausdrücke, die stets entweder true oder false als Rückgabewert liefern. Sie greifen häufig auf Vergleichsoperatoren zurück (siehe Seite 54 bis 55) und vergleichen darüber zwei Werte.

Wenn eine Variable \$logged_in den Wert true enthält, sobald ein Benutzer eingeloggt ist, und false, falls niemand eingeloggt ist, dient der folgende Ausdruck als Bedingung:

`($logged_in === true)`

Ergibt die Bedingung den Wert:

- true, könnte sie dann Anweisungen aufrufen, die einen Link zur Profilseite anzeigen.
- false, könnte sie dann Anweisungen aufrufen, die einen Link zur Login-Seite anzeigen.



IF

Eine `if`-Anweisung führt einen Code-Abschnitt nur aus, wenn eine Bedingung erfüllt ist. Die in diesem Fall zu verwendenden Anweisungen stehen in geschweiften Klammern. Ist die Bedingung nicht erfüllt, werden die Anweisungen in den geschweiften Klammern übersprungen, und der PHP-Interpreter springt zur nachfolgenden Code-Zeile.

IF... ELSE

Eine `if... else`-Anweisung überprüft eine Bedingung. Liefert diese den Wert `true`, wird die erste Gruppe von Anweisungen ausgeführt. Andernfalls wird die zweite Gruppe von Anweisungen ausgeführt.

Sie werden auch noch den ternären Operator kennenlernen, der eine Kurzform des `if... else`-Konstrukts darstellt.

IF... ELSEIF...

In einer `if... elseif`-Anweisung können Sie eine zweite Bedingung hinzufügen, die abgefragt wird, falls die erste Bedingung nicht erfüllt ist. Die auf die zweite Bedingung folgenden Anweisungen werden nur dann ausgeführt, wenn diese erfüllt ist.

Mit der `else`-Option können Sie angeben, welche Anweisungen ausgeführt werden sollen, falls keine der vorangegangenen Bedingungen erfüllt wurde.

SWITCH

Eine `switch`-Anweisung beruht nicht auf einer klassischen Bedingung; stattdessen geben Sie eine Variable vor und stellen dann unterschiedliche Werte zur Verfügung, die mit dem Variablenwert übereinstimmen könnten.

Wenn keine der angegebenen Alternativen eine Übereinstimmung liefert, kann stattdessen eine vorgegebene Reihe von Anweisungen ausgeführt werden.

Gibt es keine Übereinstimmung und keine `default`-Vorgabe, springt der PHP-Interpreter in die nächste Zeile nach der `switch`-Anweisung.

MATCH

Seit PHP 8 gibt es den `match`-Ausdruck (eine Variation des `switch`-Statements). Wird eine exakte Übereinstimmung für eine Variable gefunden (sowohl hinsichtlich des Werts als auch des Datentyps), wird ein Ausdruck ausgeführt und der durch diesen Ausdruck erzeugte Wert zurückgeliefert. Sie können mehrere Optionen pro Zeile angeben und auch eine Vorgabe für den Fall machen, dass keine übereinstimmenden Werte gefunden werden. Gibt es allerdings weder eine Übereinstimmung noch einen `Default`-Wert, wird ein Fehler ausgegeben.

```
if ($logged_in === true) {
    // für zutreffende Bedingung ausführen
}
```

Seiten 75 bis 77

```
if ($logged_in === true) {
    // für zutreffende Bedingung 1 ausführen
} elseif ($time > 12) {
    // für zutreffende Bedingung 2 ausführen
}
```

Seite 78

```
if ($logged_in === true) {
    // für zutreffende Bedingung 1 ausführen
} elseif ($time > 12) {
    // für zutreffende Bedingung 2 ausführen
} else {
    // sonst diese Anweisungen ausführen
}
```

Seite 79

```
switch ($option) {
    case 'option_1':
        // auszuführende Anweisungen
        break;
    case 'option_2':
        // auszuführende Anweisungen
        break;
    default:
        // auszuführende Anweisungen
}
```

Seite 80

```
$result = match($option) {
    'option_1'          => // Ausdruck,
    'option_2', 'option_3' => // Ausdruck,
    'default'           => // Ausdruck,
};
```

GESCHWEIFTE KLAMMERN UMSCHLIESSEN CODE-BLÖCKE

In PHP stehen zusammengehörige Anweisungen in geschweiften Klammern. Die geschweiften Klammern und die darin enthaltenen Anweisungen ergeben zusammen einen Code-Block.

ANFANG DES CODE-BLOCKS

{

```
// geschweifte Klammern  
// zeigen Anfang und Ende  
// eines Code-Blocks an.
```

ENDE DES CODE-BLOCKS

Geschweifte Klammern teilen dem PHP-Interpreter mit, wo ein Code-Block anfängt und wieder endet:

- Eine öffnende geschweifte Klammer signalisiert den Anfang eines Code-Blocks.
- Eine schließende geschweifte Klammer signalisiert das Ende eines Code-Blocks.

Es können beliebig viele Anweisungen innerhalb der geschweiften Klammern eines Code-Blocks stehen.

Codeblöcke geben dem PHP-Interpreter die Möglichkeit, die darin enthaltenen Anweisungen auszuführen, zu überspringen oder zu wiederholen.

Am Ende eines Code-Blocks sollte kein Semikolon nach der schließenden geschweiften Klammer '}' stehen, da Code-Blöcke lediglich kennzeichnen, wo eine Reihe zusammengehöriger Anweisungen beginnt und wo sie wieder endet; der Code-Block selbst ist keine Anweisung, die der PHP-Interpreter ausführt.

STRUKTUR VON BEDINGTEN ANWEISUNGEN

Eine Bedingung liefert immer einen booleschen Wert, also entweder true oder false. Das Ergebnis bestimmt, welcher Code-Block ausgeführt wird.

Die folgende Bedingung prüft, ob die Variable \$logged_in den Wert true enthält:

- Wenn ja, liefert die Bedingung den Wert true.
- Wenn nicht, liefert die Bedingung den Wert false.

ABZUFRAGENDE BEDINGUNG

```
if ($logged_in === true) {  
    $link = '<a href="member.php">My Profile</a>';  
} else {  
    $link = '<a href="login.php">Login</a>';  
}
```

AUSZUFÜHREN, WENN true

AUSZUFÜHREN, WENN false

Wenn die Bedingung erfüllt ist (true), wird der erste Code-Block ausgeführt. Der PHP-Interpreter ignoriert dann das else-Schlüsselwort und überspringt den zweiten Code-Block. Dann geht es in der ersten Code-Zelle nach der bedingten Anweisung weiter.

Wenn die Bedingung den Wert false liefert, überspringt der PHP-Interpreter den ersten Code-Block und macht beim Schlüsselwort else weiter. Er führt dann die Anweisungen im hierauf folgenden Code-Block aus. Dann geht es in der ersten Code-Zeile nach der bedingten Anweisung weiter.

IF...-STATEMENTS VERWENDEN

Dieses Beispiel zeigt eine personalisierte Begrüßung an, falls ein Nutzer eingeloggt ist. Zu Beginn werden zwei Variablen erstellt und mit Werten befüllt:

1. \$name enthält den Namen der Seitenbesucherin.
2. Die Variable \$greeting wird **initialisiert**, sie erhält also einen Anfangswert, der zum Tragen kommt, falls sie in den Schritten 3 und 4 nicht aktualisiert wird. Die Begrüßung lautet Hello.

3. Eine if-Anweisung überprüft anhand einer Bedingung, ob die Variable \$name keine leere Zeichenkette enthält.

Wenn die Variable nicht leer ist, wird der nachfolgende Code-Block ausgeführt.

4. Der Wert der Variablen \$greeting wird aktualisiert und lautet nun Welcome back, gefolgt vom Namen der Besucherin.

5. Der in \$greeting hinterlegte Wert wird auf der Seite ausgegeben.

Probieren Sie es: Ändern Sie in Schritt 1 den Wert der Variablen \$name in einen leeren String. Wenn Sie die Seite aktualisieren, lautet die Begrüßung nun Hello.

Hinweis: Eine Bedingung kann nur einen Variablenamen enthalten, zum Beispiel:

```
if ($name) {  
    $greeting = 'Hi, ' +  
    $name;  
}
```

section_a/c02/if-statement.php

PHP

```
<?php  
① $name      = 'Ivy';  
② $greeting = 'Hello';  
  
③ if ($name !== '') {  
④     $greeting = 'Welcome back, ' . $name;  
}  
?  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2><?= $greeting ?></h2>  
    </body>  
</html>
```

ERGEBNIS



Diese Bedingung überprüft, ob der in der Variablen \$name gespeicherte Wert nach erfolgter Typumwandlung als true angesehen wird.

Wie Sie auf den Seiten 60 bis 61 gesehen haben, wird eine Zeichenkette, die einen beliebigen Text oder eine Zahl ungleich 0 enthält, als boolescher Wert true behandelt.

IF... ELSE-STATEMENTS VERWENDEN

PHP

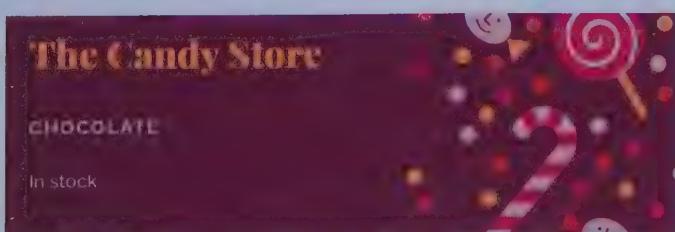
```
<?php  
① $stock = 5;  
  
② if ($stock > 0) {  
③     $message = 'In stock';  
④ } else {  
⑤     $message = 'Sold out';  
}  
?>  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Chocolate</h2>  
⑥        <p><?= $message ?></p>  
    </body>  
</html>
```

section_a/c02/if-else-statement.

Dieses Beispiel überprüft den Lagerbestand eines Artikels und zeigt eine entsprechende Meldung an.

1. Die Variable `$stock` enthält die Anzahl der vorrätigen Artikel.
2. Eine `if`-Anweisung überprüft anhand einer Bedingung, ob die in `$stock` erfasste Menge größer als 0 ist.
3. Wenn die Bedingung erfüllt ist, erhält die Variable `$message` den Wert `In stock`. Der PHP-Interpreter überspringt dann das Schlüsselwort `else` und den nachfolgenden Code-Block.
4. Wenn die Bedingung in Schritt 2 `false` zurückliefert, weist das Schlüsselwort `else` den PHP-Interpreter an, den nachfolgenden Code-Block auszuführen.
5. Die Variable `$message` erhält den Wert `Sold out`.
6. Der in der Variablen `$message` gespeicherte Wert wird auf der Seite ausgegeben.

ERGEBNIS



Probieren Sie es: Ändern Sie in Schritt 1 den in `$stock` gespeicherten Wert auf 0.

Ändern Sie die Nachricht in Schritt 5 in `More stock coming soon`.

TERNÄRE OPERATOREN

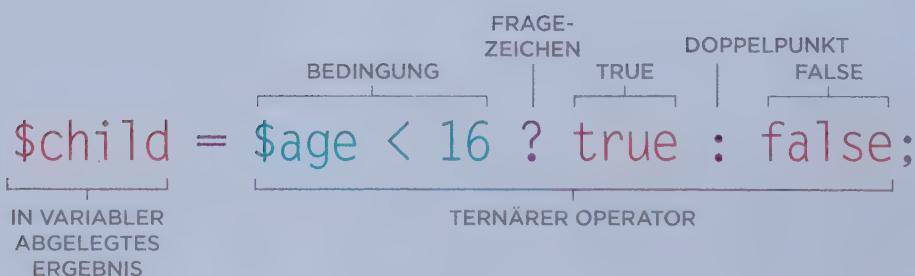
Ternäre Operatoren überprüfen eine Bedingung und geben dann einen Wert zurück, der für den Fall verwendet wird, dass die Bedingung wahr ist, und einen weiteren Wert, der für den Fall verwendet wird, dass die Bedingung unwahr ist. Sie dienen häufig als Kurzform einer `if...else`-Anweisung.

Rechts überprüft die Bedingung der `if...else`-Anweisung, ob das Alter eines Benutzers unter 16 Jahren liegt.
Ergibt die Bedingung:

- true, dann erhält `$child` den Wert `true`.
- false, dann erhält `$child` den Wert `false`.

Unten sehen Sie, wie ein ternärer Operator dies in einer einzigen Code-Zeile umsetzt.

```
if ($age < 16) {  
    $child = true;  
} else {  
    $child = false;  
}
```



Ein Fragezeichen trennt die zu überprüfende Bedingung von den zuzuweisenden Werten.

Ein Doppelpunkt trennt für den Fall einer zutreffenden Bedingung den zurückgegebenen Wert vom für den Fall einer nicht zutreffenden Bedingung zurückgegebenen Wert.

Hier wird das vom ternären Operator zurückgelieferte Ergebnis in der Variablen `$child` gespeichert.

Manchmal wird die Bedingung in Klammern gesetzt (siehe rechte Seite), um zu verdeutlichen, dass sie nur einen einzigen Ausgabewert liefert; dies ist jedoch nicht erforderlich.

TERNÄRE OPERATOREN VERWENDEN

PHP

```
<?php  
① $stock = 5;  
  
② $message = ($stock > 0) ? 'In stock' : 'Sold out';  
?>  
<!DOCTYPE html>  
<html>  
  <head> ... </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2>Chocolate</h2>  
③   <p><?= $message ?></p>  
  </body>  
</html>
```

ERGEBNIS



Dies entspricht dem vorherigen Beispiel, allerdings kommt hier ein ternärer Operator (statt einer `if ... else`-Anweisung) zum Einsatz.

1. Die Variable `$stock` enthält die Anzahl der vorrätigen Artikel.

2. Mittels eines ternären Operators wird der Variablen `$message` ein Wert zugewiesen. Die Bedingung überprüft, ob der Wert in `$stock` größer als 0 ist. Ergibt diese Bedingung:

- `true`, dann wird `In stock` in `$message` gespeichert.
- `false`, dann wird `Sold out` in `$message` gespeichert.

3. Der in der Variablen `$message` gespeicherte Wert wird auf der Seite ausgegeben.

Probieren Sie es: Ändern Sie in Schritt 1 den in `$stock` gespeicherten Wert auf 0.

Ändern Sie die Message in Schritt 2 in `More stock coming soon`.

IF... ELSEIF...-STATEMENTS VERWENDEN

Dieses Beispiel baut auf den vorhergehenden Beispielen auf.

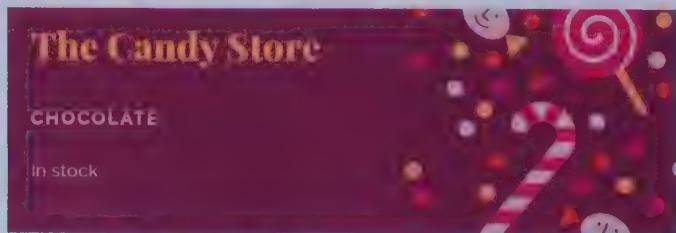
1. Die Variable \$ordered gibt die Anzahl der Artikel an, die der Shop bestellt hat, um seinen Bestand aufzufüllen.
2. Eine if-Anweisung überprüft anhand einer Bedingung, ob der Wert in \$stock größer als 0 ist. Trifft dies zu, enthält die Variable \$message den Wert In stock, und der PHP-Interpreter springt ans Ende der if... elseif-Anweisung.
3. Ist die erste Bedingung nicht erfüllt, prüft eine elseif... Anweisung anhand einer zweiten Bedingung, ob der Wert in \$ordered größer als 0 ist. Ist dies der Fall, wird der Variablen \$message der Wert Coming soon zugewiesen, und der PHP-Interpreter springt ans Ende der if... elseif-Anweisung.
4. Wenn keine der beiden Bedingungen den Wert true ergibt, führt der PHP-Interpreter die else-Bestimmung und den darauf folgenden Code-Block aus, der die Nachricht Sold out in der Variablen \$message speichert.
5. Der in \$message gespeicherte Wert wird auf der Seite ausgegeben.

section_a/c02/if-else-if-statement.php

PHP

```
<?php  
$stock    = 5;  
① $ordered = 3;  
  
② if ($stock > 0) {  
    $message = 'In stock';  
③ } elseif ($ordered > 0) {  
    $message = 'Coming soon';  
④ } else {  
    $message = 'Sold out';  
}  
?  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Chocolate</h2>  
⑤         <p><?= $message ?></p>  
    </body>  
</html>
```

ERGEBNIS



Probieren Sie es: Ändern Sie in Schritt 1 den Wert der Variablen \$stock auf 0.

Beim Aktualisieren der Seite sollte die Meldung Coming soon erscheinen.

SWITCH-STATEMENTS VERWENDEN

PHP

section_a/c02/switch-statement.php

```
<?php  
① $day = 'Monday';  
  
② switch ($day) {  
    ③     case 'Monday':  
        ④         $offer = '20% off chocolates';  
        ⑤         break;  
    ③     case 'Tuesday':  
        ④         $offer = '20% off mints';  
        ⑤         break;  
    ⑥     default:  
        ⑦         $offer = 'Buy three packs, get one free';  
    }  
?>  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
        <h2>Offers on <?= $day; ?></h2>  
        <p><?= $offer ?></p>  
    </body>  
</html>
```

ERGEBNIS



1. Eine Variable \$day wird eingerichtet, um einen Wochentag zu speichern.

2. Die switch-Anweisung beginnt mit dem Befehl switch und einem eingeklammerten Variablenamen. Diese Variable enthält den sogenannten switch-Wert.

Danach folgt ein geschweiftes Klammerpaar mit möglichen Optionen, die mit dem switch-Wert übereinstimmen könnten.

3. Hier gibt es zwei Optionen. Beide:

- beginnen mit dem Wort case
- gefolgt von einem Wert
- und dann einem Doppelpunkt

4. Wenn der switch-Wert mit einer der Optionen übereinstimmt, werden die darauf folgenden Anweisungen ausgeführt. (Diese setzen Werte für die Variable \$offer.)

5. break veranlasst den PHP-Interpreter, ans Ende der switch-Anweisung zu springen.

6. Die letzte Option nennt sich default. Ihr folgen die Anweisungen, die ausgeführt werden sollen, wenn keine der vorherigen Optionen zutrifft. (Auf der Option default folgt kein break.)

7. Der Wert in \$offer wird angezeigt.

Probieren Sie es: Ändern Sie den Wert in Schritt 1 auf Wednesday. Fügen Sie der switch-Anweisung nach Schritt 5 noch eine weitere Option für Wednesday hinzu.

MATCH-AUSDRÜCKE VERWENDEN

Anmerkung: Dieses Beispiel funktioniert nur mit PHP 8 und aufwärts.

1. Eine Variable \$day wird eingerichtet, um einen Wochentag zu speichern.

2. Mittels eines match-Ausdrucks wird der Variablen \$offer ein Wert zugewiesen. Dieser beginnt mit dem Wort match, gefolgt vom eingeklammerten Variablennamen und einer öffnenden geschweiften Klammer.

3. Die geschweiften Klammern enthalten mehrere Zweige, die mit Werten beginnen, die auf ihre Übereinstimmung mit dem in der Variablen \$day gespeicherten Wert überprüft werden.

Bei einer Übereinstimmung wird der rechts vom Doppelpfeil-Operator stehende Ausdruck ausgeführt.

Jeder Zweig kann nur einen einzelnen Ausdruck ausführen und endet mit einem Komma. Beachten Sie auch, dass der match-Ausdruck einen strikten Typvergleich verwendet; es wird also kein automatisches Typecasting durchgeführt.

4. Der letzte Zweig nutzt den Wert default, gefolgt von einem Ausdruck, der ausgeführt wird, wenn es ansonsten keine Übereinstimmungen gibt. (Wenn es keine Übereinstimmung und keinen default-Zweig gibt, wird ein Fehler ausgeworfen).

5. Der Wert in \$offer wird angezeigt.

section_a/c02/match.php

```
<?php  
① $day = 'Monday';  
  
② $offer = match($day) {  
    'Monday'          => '20% off chocolates',  
    ③ 'Saturday', 'Sunday' => '20% off mints',  
    ④ default         => '10% off your entire order',  
}  
?  
<!DOCTYPE html>  
<html>  
  <head> ... </head>  
  <body>  
    <h1>The Candy Store</h1>  
    <h2>Offers on <?= $day ?></h2>  
    <p><?= $offer ?></p>  
  </body>  
</html>
```

PHP

ERGEBNIS



Probieren Sie es: Ändern Sie den Wert in Schritt 1 in Tuesday. Fügen Sie dem match - Ausdruck in Schritt 3 ein entsprechendes Dienstagsangebot hinzu.

Probieren Sie es: Ändern Sie den Wert in Schritt 1 in Wednesday. Entfernen Sie dann die default-Option. Jetzt sollten Sie eine Fehlermeldung erhalten.

SCHLEIFEN

Mit einer Schleife können Sie eine Abfolge von Anweisungen einmal schreiben und diese dann entweder eine festgelegte Anzahl von Wiederholungen oder bis zur Erfüllung einer Bedingung durchlaufen lassen.

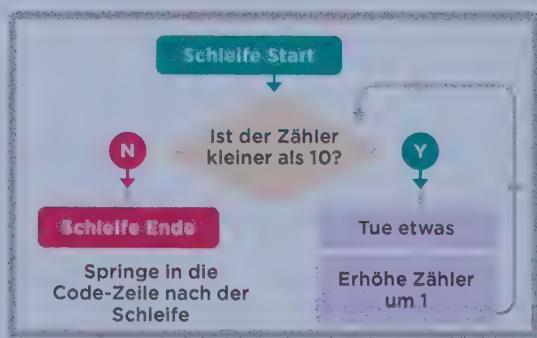
Wenn Sie möchten, dass eine Person dieselbe Aufgabe zehnmal erledigt, brauchen Sie nicht zehnmal dieselben Anweisungen aufzuschreiben; stattdessen genügt es, die Instruktionen einmal aufzuschreiben und der Person zu sagen, dass sie die Aufgabe zehnmal wiederholen soll. In PHP gelingt Ihnen das mit Schleifen:

- Schreiben Sie die Statements zur Erfüllung einer Aufgabe einmal in geschweifte Klammern, sodass sie einen Code-Block bilden.
- Stellen Sie anhand einer Bedingung fest, ob diese Anweisungen ausgeführt werden sollen oder nicht (zum Beispiel mit der `if`-Anweisung auf Seite 74). Wenn die Bedingung `true` zurückgibt, wird der Code-Block ausgeführt; gibt sie den Wert `false` zurück, wird er nicht ausgeführt.
- Nachdem die Anweisungen ausgeführt wurden, wird die Bedingung erneut überprüft. Wenn sie den Wert `true` ergibt, werden die Anweisungen nochmals ausgeführt und anschließend die Bedingung erneut überprüft.

Wenn die Bedingung `false` zurückgibt, springt der Interpreter in die Code-Zeile nach der Schleife.

Um eine Aufgabe zehnmal auszuführen, könnten Sie eine Variable als Zähler verwenden, ihr den Wert 1 geben.

1. Dann überprüfen Sie, ob der Wert des Zählers kleiner als 10 ist.
2. Ist dies der Fall, werden die Anweisungen im Code-Block ausgeführt.
3. Der im Zähler gespeicherte Wert wird um 1 erhöht.
4. Der PHP-Interpreter kehrt zu Schritt 1 zurück.



WHILE

Seite 82 bis 83

Eine `while`-Schleife wiederholt die darin enthaltenen Anweisungen, solange eine Bedingung den Wert `true` liefert.

FOR-SCHLEIFEN

Seite 86 bis 89

Mit einer `for`-Schleife können Sie eine bestimmte Anzahl von Durchläufen für den Code-Block festlegen. Auf die Bedingung folgen Anweisungen, die einen Zähler initialisieren und dann bei jedem Schleifendurchlauf aktualisieren.

DO WHILE-SCHLEIFEN

Seite 84 bis 85

Eine `do ... while`-Schleife ist wie eine `while`-Schleife, wobei die Bedingung allerdings erst nach der Ausführung überprüft wird. Die Anweisungen werden also stets mindestens einmal durchlaufen, auch wenn sich die Bedingung später als unwahr herausstellt.

FOREACH-SCHLEIFEN

Seite 90 bis 93

Eine `foreach`-Schleife durchläuft sämtliche Elemente eines Arrays und wiederholt für jedes Element die gleiche Abfolge von Anweisungen. (Sie kann auch mit Objekteigenschaften arbeiten, die Sie in Kapitel 4 kennenlernen.)

WHILE-SCHLEIFEN

Die `while`-Schleife überprüft eine Bedingung; ist diese erfüllt, wird ein Code-Block ausgeführt. Die Bedingung wird anschließend erneut geprüft; ist sie immer noch erfüllt, wird der Code-Block erneut ausgeführt. Die Schleife wird also so oft wiederholt, bis die Bedingung nicht mehr erfüllt ist.

SCHLEIFENTYP

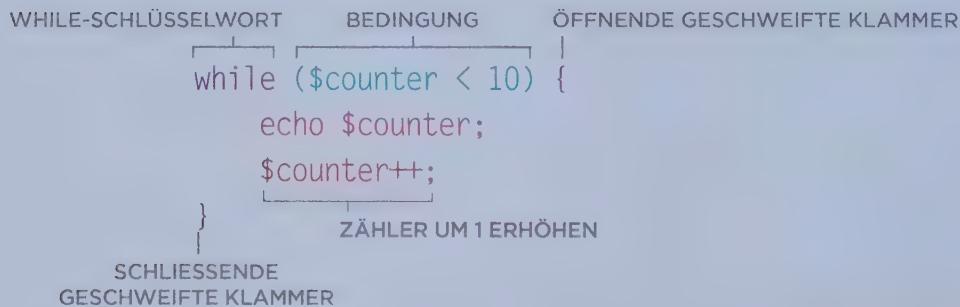
Alle Schleifen beginnen mit einem Schlüsselwort, das dem PHP-Interpreter mitteilt, um welche Art von Schleife es sich handelt. Eine `while`-Schleife beginnt mit dem Schlüsselwort `while`.

BEDINGUNG

Die Bedingung überprüft Werte im Code. (Die folgende Bedingung prüft, ob der Wert der Variablen `$counter` kleiner als 10 ist.) Falls die Bedingung den Wert `true` annimmt, werden die Anweisungen in geschweiften Klammern ausgeführt.

AUSZUFÜHRENDE ANWEISUNGEN

Die Anweisungen zur Durchführung der sich wiederholenden Aufgabe stehen in geschweiften Klammern. Schleifen durchlaufen den Code in diesen geschweiften Klammern immer wieder, bis die Bedingung nicht mehr erfüllt ist.



Das obige Beispiel besagt:

Solange der Wert der Variablen `$counter` kleiner als 10 ist, wiederhole die Anweisungen in den geschweiften Klammern.

Und der Code in den geschweiften Klammern:

1. Gibt den in der Variablen `$counter` gespeicherten Wert aus
2. Erhöht mit dem Operator `++` den Wert der Variablen `$counter` um 1

Falls `$counter` beim Einstieg in diesen Code den Wert 1 besitzt, wird 123456789 angezeigt (da der Inhalt von `$counter` ausgegeben wird, bis er den Wert 10 erreicht).

WHILE-SCHLEIFEN VERWENDEN

PHP

```
section_a/c02/while-loop.php

<?php
① $counter = 1;
② $packs   = 5;
③ $price   = 1.99;
?>

...
<h2>Prices for Multiple Packs</h2>
<p>
    <?php
④ while ($counter <= $packs) {
    echo $counter;
    echo ' packs cost $';
    echo $price * $counter;
    echo '<br>';
    $counter++;
}
?>
</p>
```

ERGEBNIS



Probieren Sie es: Erhöhen Sie in Schritt 2 die Anzahl der Päckchen auf 10.

Probieren Sie es: Ändern Sie in Schritt 4 den Operator von \leq in $<$.

Dieses Beispiel zeigt den Preis für mehrere Packungen Süßigkeiten an.

1. Die Variable `$counter` erhält den Startwert 1.
 2. `$packs` enthält die Anzahl der Packungen, für die die Preise angezeigt werden sollen.
 3. `$price` ist der Preis pro Packung.
 4. Die `while`-Schleife fängt mit einer Bedingung an. Diese überprüft, ob der Wert von `$counter` kleiner oder gleich dem in `$packs` gespeicherten Wert ist. Trifft dies zu, werden die Anweisungen in den geschweiften Klammern ausgeführt.
 5. Die Zahl in der Zählvariablen `$counter` wird ausgegeben.
 6. Der Text `packs cost $` wird angezeigt (auf Seite 100 sehen Sie, wie Sie die Einzahl `pack` für ein einzelnes Päckchen verwenden).
 7. Der Wert in `$price` wird mit dem Wert in `$counter` multipliziert und ausgegeben.
 8. Ein Zeilenumbruch wird eingefügt.
 9. Der Wert von `$counter` wird mit dem Inkrement-Operator um 1 erhöht (siehe Seite 50).
- Nach Schritt 9 überprüft der PHP-Interpreter erneut die Bedingung aus Schritt 4. Dieser Vorgang wird so lange wiederholt, bis diese Bedingung `false` ergibt.

DO WHILE-SCHLEIFEN

Eine do while-Schleife führt zunächst eine Reihe von Anweisungen in geschweiften Klammern aus. Erst dann wird die Bedingung überprüft, sodass der Code-Block immer mindestens einmal ausgeführt wird.

SCHLEIFENTYP

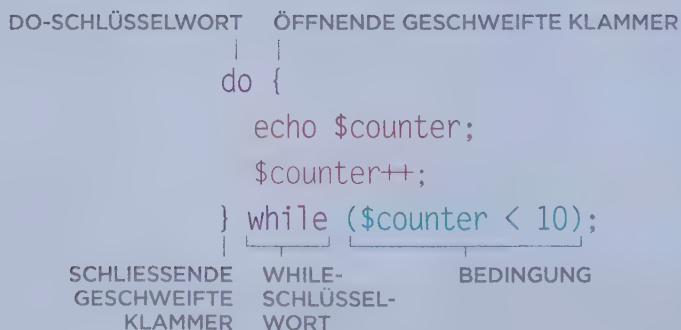
Eine do while-Schleife beginnt mit dem Schlüsselwort `do`. Das Schlüsselwort `while` steht dabei nach der schließenden geschweiften Klammer, die die auszuführenden Anweisungen enthält.

AUSZUFÜHRENDE ANWEISUNGEN

Die Anweisungen zur Durchführung der sich wiederholenden Aufgabe stehen in geschweiften Klammern. Sie werden mindestens einmal ausgeführt, da die Bedingung erst hinter den geschweiften Klammern steht.

BEDINGUNG

Die Bedingung überprüft die aktuellen Werte im Code. Falls sie den Wert `true` ergibt, kehrt der PHP-Interpreter an den Anfang der Schleife zurück und wiederholt die Anweisungen.



Die Anweisungen in dieser Schleife geben den Wert der Variablen `$counter` aus und erhöhen diesen dann mit dem Operator `++` um den Wert 1.

Die Anweisungen werden noch vor der Bedingung ausgeführt, es wird also stets der Wert in `$counter` ausgegeben und mindestens einmal der Wert 1 dazu addiert.

Falls `$counter` beim Einstieg den Wert 3 besitzt, gibt dieser Code 3456789 aus. Hat `$counter` den Wert 1, gibt er 123456789 aus.

DO WHILE-SCHLEIFEN VERWENDEN

PHP

section_a/c02/do-while-loop.php

```
<?php  
① [ $packs = 5;  
    $price = 1.99;  
    ?>  
...  
<h2>Prices for Multiple Packs</h2>  
<p>  
    <?php  
② do {  
    ③ [ echo $packs;  
        echo ' packs cost $';  
        echo $price * $packs;  
        echo '<br>';  
    ④ [ $packs--;  
    ⑤ } while ($packs > 0);  
    ?>  
</p>
```

ERGEBNIS



Probieren Sie es: Ändern Sie in Schritt 1 den Wert für \$packs auf 10 und den Wert für \$price auf 2.99.

Hinweis: Nach der schließenden geschweiften Klammer (vor dem Schlüsselwort while) steht kein Semikolon, nach der Bedingung dagegen schon.

In diesem Beispiel wird der in geschweiften Klammern stehende Code noch vor Überprüfung der Bedingung abgearbeitet, sodass der Code-Block mindestens einmal ausgeführt wird, auch wenn die Bedingung nicht erfüllt ist.

1. Es werden zwei Variablen eingerichtet. Die Anzahl der Süßigkeitenpackungen wird in \$packs gespeichert. Der Preis pro Päckchen wird in \$price gespeichert.
2. Die do while-Schleife beginnt mit dem Schlüsselwort do und einer öffnenden geschweiften Klammer. Der Code-Block steht vor der Bedingung, wird also immer mindestens einmal ausgeführt, egal ob die Bedingung erfüllt ist oder nicht.
3. Die Anzahl der Päckchen (gespeichert in \$packs) gefolgt von der Zeichenkette packs cost \$ wird ausgegeben (auf Seite 100 sehen Sie, wie Sie die Einzahl pack für ein einzelnes Päckchen verwenden).
4. Die Preise werden ausgegeben, indem \$packs mit \$price multipliziert wird. Es folgt ein Zeilenumbruch.
5. Der in \$packs gespeicherte Wert wird mit dem Dekrement-Operator -- um eins verringert.
6. Der Code-Block endet mit einer schließenden geschweiften Klammer. Es folgt das Schlüsselwort while, dann die Bedingung. Die Bedingung fragt ab, ob die in \$packs gespeicherte Zahl größer als 0 ist.

FOR-SCHLEIFEN

Die Anweisungen in einer for-Schleife durchlaufen eine vorgegebene Anzahl von Wiederholungen. Hierzu wird ein Zähler erstellt und bei jedem Schleifendurchlauf aktualisiert.

SCHLEIFENTYP

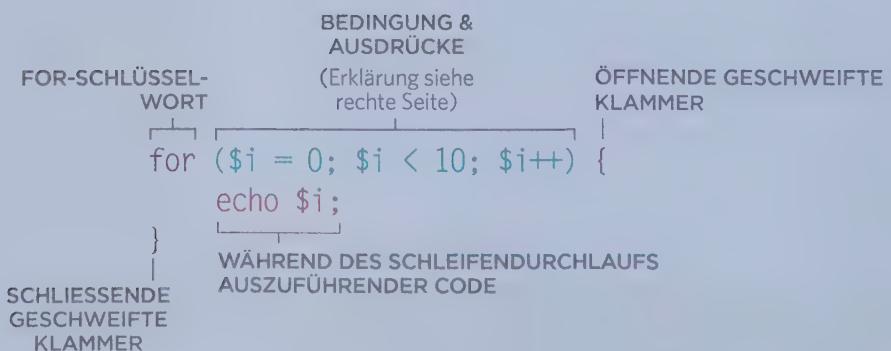
Eine for-Schleife beginnt mit dem Schlüsselwort for.

BEDINGUNG

Die Bedingung einer for-Schleife steht neben dem Code zur Initialisierung und Aktualisierung des Zählers. Auf der rechten Seite stellen wir dies im Detail dar.

AUSZUFÜHRENDE ANWEISUNGEN

Die Anweisungen zur Durchführung der zu wiederholenden Aufgabe stehen in geschweiften Klammern. Die Anzahl der Durchläufe ist festgelegt.



Für den Zähler wird oft der Variablenname \$i oder \$index verwendet.

Die Anweisung in den geschweiften Klammern gibt den in \$i gespeicherten Wert aus.

In diesem Fall lautet die Ausgabe:
0123456789.

FOR-SCHLEIFEN VERWENDEN DREI AUSDRÜCKE

Die for-Schleife braucht neben der Bedingung noch zwei weitere Ausdrücke, einen zur Erstellung eines Zählers und einen zu dessen Aktualisierung.

AUSDRUCK 1:

INITIALISIERUNG

Dieser Ausdruck wird nur einmal ausgeführt, nämlich beim ersten Schleifendurchlauf. Er erstellt die Variable für den Zähler und setzt ihren Wert auf 0.

AUSDRUCK 2:

BEDINGUNG

Beim zweiten Ausdruck handelt es sich um die Bedingung. Die Anweisungen in der `for`-Schleife werden so lange wiederholt, bis die Bedingung `false` zurückliefert.

AUSDRUCK 3:

AKTUALISIERUNG

Nachdem die Anweisungen in den geschweiften Klammern ausgeführt wurden, addiert der dritte Ausdruck den Wert 1 zu der im Zähler gespeicherten Zahl.

INITIALISIERUNG BEDINGUNG AKTUALISIERUNG
(\$i = 0; \$i < 10; \$i++)

In dem obigen Beispiel dient die Variable `$i` als Zähler. Sie erhält einen Anfangswert von 0.

Die Bedingung fragt ab, ob der Wert von `$i` kleiner als 10 ist. Solange dies zutrifft, werden die Anweisungen im Code-Block ausgeführt.

Anstelle der Zahl 10 könnte es sich auch um eine Variable handeln, die einen Wert enthält, z.B. $\$i < \max .

Nach jedem Schleifendurchlauf wird der Zähler mit dem Inkrement-Operator `++` aktualisiert, um den in `$i` gespeicherten Wert um 1 zu erhöhen.

FOR-SCHLEIFEN VERWENDEN

In diesem Beispiel wird eine Aufgabe mithilfe einer for-Schleife zehnmal wiederholt.

1. Die Variable \$price speichert die Kosten für ein einzelnes Päckchen Süßigkeiten.

2. Das Schlüsselwort for gibt den Schleifentyp an, danach folgen die drei Ausdrücke in Klammern:

- Erster Ausdruck: `$i = 1;`
Hier wird der Zähler mit dem Wert 1 initialisiert.

- Zweiter Ausdruck: `$i <= 10;`
Dies ist die Bedingung. Sie besagt, dass der Code wiederholt werden soll, solange der Zählerwert kleiner als oder gleich 10 ist.

- Dritter Ausdruck: `$i++`
Hierdurch wird der Wert des Zählers bei jedem Schleifendurchlauf um 1 erhöht.

3. Die geschweiften Klammern enthalten die Anweisungen, die bei jedem Schleifendurchlauf abgearbeitet werden. Wie zuvor werden die Packungsanzahl (der im Zähler gespeicherte Wert) und der Preis (der Zählerwert multipliziert mit dem Wert in \$price) ausgegeben.

Nachdem die Anweisungen in den geschweiften Klammern ausgeführt wurden, aktualisiert der dritte Ausdruck (in Schritt 2) den Zähler, indem er den in \$i gespeicherten Wert um 1 erhöht.

section_a/c02/for-loop.php

PHP

```
<?php  
① $price = 1.99;  
?>  
...  
<h2>Prices for Multiple Packs</h2>  
<p>  
② <?php  
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
    echo ' packs cost '$;  
    echo $price * $i;  
    echo '<br>';  
}  
?>  
</p>
```

ERGEBNIS



Probieren Sie es: Erhöhen Sie den Preis in Schritt 1 von 1.99 auf 2.99.

Probieren Sie es: Sorgen Sie in Schritt 2 dafür, dass die Schleife 20-mal durchlaufen wird.

PHP

section_a/c02/for-loop-higher-counter.php

```

<?php
① $price = 1.99;
?>

...
<h2>Prices for Large Orders</h2>
<p>
<?php
② for ($i = 10; $i <= 100; $i = $i + 10) {
    echo $i;
    echo ' packs cost $';
    echo $price * $i;
    echo '<br>';
}
?>
</p>
```

ERGEBNIS

Auf Seite 216 erfahren Sie, wie Sie Zahlen so formatieren, dass sie bei der Ausgabe zwei Dezimalstellen aufweisen (so wie wir es von Preisen her gewohnt sind).

Dieses Beispiel zeigt Preisnachlässe für den Kauf mehrerer Süßwarenpackungen an.

1. Eine Variable speichert den Preis für ein einzelnes Päckchen Süßigkeiten.
2. Das Schlüsselwort `for` gibt den Schleifentyp an, danach folgen die drei Ausdrücke in Klammern:
 - Erster Ausdruck: `$i = 10`; Hier wird der Zähler mit dem Wert 10 initialisiert.
 - Zweiter Ausdruck: `$i <= 100`; Dies ist die Bedingung. Sie besagt, dass der Code wiederholt werden soll, solange der Zählerwert kleiner als oder gleich 100 ist.
 - Dritter Ausdruck: `$i = $i + 10`; Hierdurch wird der Wert des Zählers bei jedem Schleifendurchlauf um 10 erhöht.
3. Die geschweiften Klammern enthalten die Anweisungen, die bei jedem Schleifendurchlauf abgearbeitet werden. Sie entsprechen denen aus dem vorhergehenden Beispiel.

Verändern Sie in Schritt 2 die Bedingung so, dass die Preise für bis zu 200 Päckchen Süßigkeiten angezeigt werden.

FOREACH-SCHLEIFEN

Eine `foreach`-Schleife ist für die Verarbeitung von zusammengesetzten Datentypen wie Arrays konzipiert. Sie durchläuft jedes Feldelement der Reihe nach und führt dabei jeweils denselben Code-Block aus.

Zusammengesetzte Datentypen wie Arrays enthalten eine Reihe zusammengehöriger Elemente. Jedes Element besteht aus einem Schlüssel/Wert-Paar. In assoziativen Arrays ist der Schlüssel eine Zeichenkette, in indizierten Arrays übernimmt diese Funktion eine Indexnummer.

Eine `foreach`-Schleife arbeitet jedes Element in einem Array einzeln ab. Sie führt die Anweisungen im Code-Block aus und wechselt dann zum nächsten Element.

Bei jeder Ausführung des Code-Blocks können Sie auf den Schlüssel und den Wert des aktuellen Array-Elements zugreifen und diese innerhalb des Code-Blocks verwenden. Dazu geben Sie in den Klammern nach dem Schlüsselwort `foreach` Folgendes an:

- den Namen der Variablen, die das Array enthält
- einen Variablenamen, der den aktuellen Schlüssel aufnimmt
- einen Variablenamen, der den aktuellen Wert aufnimmt

SCHLEIFENTYP

Eine `foreach`-Schleife beginnt mit dem Schlüsselwort `foreach`.

VARIABLENNAMEN

Die Klammern enthalten drei Variablennamen (die auf der rechten Seite ausführlich behandelt werden).

AUSZUFÜHRENDE ANWEISUNGEN

Die Anweisungen zur Durchführung der sich wiederholenden Aufgabe stehen in geschweiften Klammern.

VARIABLE, DIE DAS ARRAY ENTHÄLT

Schlüsselwort

```
foreach ($array as $key => $value) {  
    echo $key;  
    echo ' - '$;  
    echo $value;  
}
```

VARIABLENNAME FÜR SCHLÜSSEL

VARIABLENNAME FÜR WERT

In den geschweiften Klammern stehen drei Anweisungen, die Folgendes ausgeben:

- den Schlüssel
- einen Bindestrich und ein Dollarzeichen
- den Wert des Elements

Wenn Sie nur die Werte des Arrays verwenden wollen, können Sie den Schlüssel weglassen:

```
foreach ($array as $value) {  
    // Anweisungen hier rein  
}
```

Sobald dieselben Anweisungen für jedes einzelne Element des Arrays wiederholt wurden, wechselt der PHP-Interpreter in die nächste Code-Zeile nach der Schleife.

Unten speichert \$products eine Reihe von Produktbezeichnungen und -preisen.

Eine foreach-Schleife kann für jeden Artikel den Namen und den Preis anzeigen.

Die Schleife funktioniert unabhängig von der Anzahl der Elemente im Array.

`$products = ['toffee' => 2.99, 'mints' => 1.99, 'fudge' => 3.40,];`

VARIABLE	SCHLÜSSEL	WERT	SCHLÜSSEL	WERT	SCHLÜSSEL	WERT
----------	-----------	------	-----------	------	-----------	------

Im folgenden Beispiel beginnt die Schleife mit dem Schlüsselwort foreach.

Dann steht in Klammern:

- \$products, der Name der Variablen, die das Array enthält
- gefolgt vom Schlüsselwort as

- die Variable \$item enthält den Schlüssel des aktuellen Feld-elements
- darauf folgt ein Doppelpfeil-Operator
- die Variable \$price enthält den Wert des aktuellen Feld-elements

Im Code-Block stehen die Variablennamen \$item und \$price für den Schlüssel beziehungsweise den Wert des aktuellen Array-Elements. Zuerst wird der Produktname (gespeichert in \$item) ausgegeben, gefolgt von einem Dollarzeichen und einem Bindestrich und schließlich dem Preis (gespeichert in \$price).

SCHLÜSSELWORT AS **DOPPELPFEIL-OPERATOR**

VARIABLE, DIE DAS ARRAY ENTHÄLT	VARIABLENNAME FÜR SCHLÜSSEL	VARIABLENNNAME FÜR WERT
---------------------------------	-----------------------------	-------------------------

```
foreach ($products as $item => $price) {  
    echo $item;  
    echo ' - '$;  
    echo $price;  
}
```

Schleifen werden häufig im HTML-erzeugenden Seitenteil verwendet (siehe nächste Seite).

In diesem Fall können die erste und die letzte Zeile der obigen Schleife in eigene Code-Blöcke gesetzt werden.

Die Daten in den Schlüsseln und Werten des Arrays lassen sich dann mit der Kurzschreibweise für echo ausgeben.

```
<?php foreach ($products as $item => $price) { ?>  
    <li>  
        <b><?= $item ?></b> - $<?= $price ?>  
    </li>  
<?php } ?>
```

SCHLÜSSEL UND WERTE PER SCHLEIFE DURCHLAUFEN

Dieses Beispiel gibt eine Tabelle mit den Namen und Preisen der in einem Array gespeicherten Produkte aus.

1. Die Variable \$products enthält ein assoziatives Array mit einer Liste von Produkten und deren Preisen.
2. Im HTML-Teil der Seite werden eine Überschrift und der Anfang der HTML-Tabelle ausgegeben.
3. Eine foreach-Schleife wird eingerichtet.

In den Klammern nach dem Schlüsselwort foreach sehen Sie:

- \$products - den Namen der Variablen, die das Array enthält
- das Schlüsselwort as
- \$item - ein Variablenname, der für den Schlüssel des aktuellen Array-Elements steht
- den Doppelpfeil-Operator
- \$price - ein Variablenname, der für den Wert des aktuellen Array-Elements steht

Es folgt eine öffnende geschweifte Klammer, mit der ein Code-Block beginnt.

4. Für jedes Feldelement wird eine Tabellenzeile mit dem Namen und dem Preis ausgegeben.
5. Der Code-Block wird abgeschlossen.

Probieren Sie es: Fügen Sie in Schritt 1 zwei weitere Elemente in das Array ein.

section_a/c02/foreach-loop.php

PHP

```
<?php  
$products = [  
    'Toffee' => 2.99,  
    'Mints'  => 1.99,  
    'Fudge'  => 3.49,  
];  
?  
...  
<h2>Price List</h2>  
<table>  
    <tr>  
        <th>Item</th>  
        <th>Price</th>  
    </tr>  
    <?php foreach ($products as $item => $price) { ?>  
        <tr>  
            <td><?= $item ?></td>  
            <td>$<?= $price ?></td>  
        </tr>  
    <?php } ?>  
</table> ...
```

ERGEBNIS



PHP section_a/c02/foreach-loop-just-accessing-values.php

```
<?php  
① $best_sellers = ['Toffee', 'Mints', 'Fudge',];  
?>  
...  
② <h2>Best Sellers</h2>  
③ <?php foreach ($best_sellers as $product) { ?>  
④   <p><?= $product ?></p>  
⑤ <?php } ?>
```

ERGEBNIS



In einem indizierten Array stellen die Indexnummern oft die Reihenfolge der Feldelemente dar. Über eine `foreach`-Schleife lassen sich die Werte in dieser Reihenfolge ausgeben. WIR LERNEN DAS

1. Die Variable `$best_sellers` enthält ein indiziertes Array mit den meistverkauften Produkten.

2. Mit einer `foreach`-Schleife werden die meistverkauften Produkte angezeigt. Nach dem Schlüsselwort `foreach` steht in Klammern:

- `$best_sellers` - der Name der Variablen, die das Array enthält
- Das Schlüsselwort `as`
- `$product` - ein Variablenname, der für den Wert des aktuellen Array-Elements steht
- es gibt keine Variable für den Schlüsselnamen (die Indexnummer)

Darauf folgt eine öffnende geschweifte Klammer, die den Code-Block einleitet.

3. Der Wert des aktuellen Elements wird in `<p>`-Tags ausgegeben. WIR LERNEN DAS

4. Der Code-Block wird abgeschlossen.

Probieren Sie es: Fügen Sie in Schritt 1 zwei weitere Elemente in das Array ein. Ändern Sie dann in den Schritten 2 und 3 den Variablennamen von `$product` in `$candy`.

INCLUDE-DATEIEN ZUR WIEDERHOLUNG VON CODE EINBINDEN

Für die meisten Websites ist es erforderlich, identischen Code auf mehreren Seiten zu wiederholen. Zum Beispiel sind die Kopf- und Fußbereiche oft auf allen Seiten identisch. Dank der Include-Dateien brauchen Sie die gleichen Code-Zeilen dabei nicht in mehreren Dateien zu wiederholen.

Statt den Code für den Website-Header auf jeder Unterseite zu duplizieren, können Sie:

- den Code für den Header in eine separate PHP-Datei auslagern, die als Include File bezeichnet wird.
- über die `include`-Anweisung von PHP diesen Code in jede Seite einzufügen, die den Header benötigt.

Wenn der PHP-Interpreter auf eine Include-Anweisung stößt, ruft er den Inhalt der Include-Datei ab und führt den Code so aus, als stünde er an der Stelle der Include-Anweisung.

Die Datei `candy.php` unten links bindet zwei Dateien ein:

- `header.php` enthält den Website-Header.
- `footer.php` enthält den Website-Footer.

Zwischen den beiden Include-Anweisungen steht der Hauptinhalt der Seite. Mithilfe von Include-Dateien:

- sparen Sie es sich, denselben Code mehrfach zu kopieren oder zu wiederholen.
- wird die Pflege des Codes vereinfacht, da sich die Änderung einer Include-Datei auf alle Seiten auswirkt, die darauf zurückgreifen.

`candy.php`

```
<?php include 'includes/header.php'; ?>

<h1>The Candy Store</h1>
<h2>Welcome</h2>
<p>A wide selection of delicious candy
handmade in our kitchen...</p>

<?php include 'includes/footer.php'; ?>
```

`includes/header.php`

```
<h1>The Candy Store</h1>
<nav>
  <a href="index.php">Home</a> |
  <a href="candy.php">Candy</a> |
  <a href="about.php">About</a> |
  <a href="contact.php">Contact</a> |
</nav>
```

`includes/footer.php`

```
<footer>
  &copy; <?php echo date('Y')?>
</footer>
```

DATEIEN EINBINDEN ÜBER INCLUDE UND REQUIRE

Es gibt insgesamt vier Schlüsselwörter, um Code aus einer Include-Datei einzubinden. Jedes davon verhält sich etwas anders, bedient sich aber der gleichen Syntax.

Das Schlüsselwort `include` weist den PHP-Interpreter an, eine zusätzliche Datei vom Server zu laden und sie so zu behandeln, als stünde ihr Inhalt anstelle der `include`-Anweisung.

Darauf folgt der in Anführungszeichen gesetzte Dateipfad. (Manchmal werden der Dateiname und die Anführungszeichen auch eingeklammert, aber das ist nicht nötig). Die Include-Datei sollte die Dateiendung `.php` tragen.

INCLUDE-ANWEISUNG
`<?php include 'includes/filename.php'; ?>`
RELATIVER DATEIPFAD

include / require

Die Schlüsselwörter `include` und `require` binden beide den Code aus der externen Datei ein, deren Pfad auf das Schlüsselwort folgt.

Allerdings verhält sich der PHP-Interpreter in beiden Fällen unterschiedlich, falls die einzubindende Datei nicht gefunden oder gelesen werden kann:

- `include`: Der Interpreter liefert einen Fehler, versucht aber, den Rest der Seite zu verarbeiten.
- `require`: Der Interpreter liefert einen Fehler und bricht die Verarbeitung der restlichen Seite ab.

include_once / require_once

Die Schlüsselwörter `include_once` und `require_once` leisten genau dasselbe wie `include` und `require`, stellen dabei aber sicher, dass der PHP-Interpreter den Code nur einmal in eine bestimmte Seite einfügt.

Ist eine Datei einmal mit den Schlüsselwörtern `include_once` und `require_once` in die Seite eingebunden worden, wird sie kein zweites Mal eingebunden, wenn die Seite mit denselben Schlüsselwörtern versucht, die gleiche Datei einzubinden.

Der PHP-Interpreter benötigt zusätzliche Ressourcen, um zu prüfen, ob die Datei bereits eingebunden wurde. Diese Anweisungen sollten also nur dann verwendet werden, wenn die Gefahr einer doppelten Einbindung besteht.

INCLUDE-DATEIEN ANLEGEN

Diese Seite zeigt zwei Include-Dateien:

- header.php enthält das anfängliche HTML-Markup, den Namen der Website und die Navigationselemente, die oben auf jeder Seite der Website angezeigt werden.
- footer.php enthält einen Urheberrechtshinweis mit dem aktuellen Jahr und den abschließenden HTML-Code für jede Seite.

Beide Dateien nutzen die Endung .php. Damit wird gewährleistet, dass der PHP-Code vom PHP-Interpreter ausgeführt wird.

Include Files werden häufig in einem Ordner includes gespeichert (so wie diese beiden Dateien).

Dieses Beispiel zeigt, dass Sie zur Aktualisierung der Seiten-navigation nur die Datei header.php anpassen müssen. Jede Seite, die diese Datei einbin-det, wird dann auch automatisch aktualisiert.

Hinweis: Die Links in header.php dienen zur Ver-anschaulichung einer Naviga-tionsleiste; die Code-Beispiele im Downloadbereich enthalten keine entsprechenden Unterseiten für die einzelnen Links.

section_a/c02/includes/header.php

PHP

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css"
  />
  </head>
  <body>
    <h1>The Candy Store</h1>
    <nav>
      <a href="index.php">Home</a> |
      <a href="candy.php">Candy</a> |
      <a href="about.php">About</a> |
      <a href="contact.php">Contact</a>
    </nav>
```

section_a/c02/includes/footer.php

PHP

```
<footer>&copy; <?php echo date('Y')?></footer>
</body>
</html>
```

Wenn eine Include-Datei in der letzten Zeile mit einer PHP-Anweisung endet, fehlt oft das abschließende ?>-PHP-Tag, weil Leerzeichen nach einem abschlie-ßenden Tag zu unerwünschtem Leerraum im Browser führen können. Es könnte auch dafür sorgen, dass HTTP-Header (siehe die Seiten 180 bis 182) zu früh an den Browser gesendet werden.

Möglicherweise sehen Sie auch eine Leerzeile am Ende der Include-Datei. Diese wird manch-mal eingefügt, damit bestimmte Tools die Änderungen zwischen verschiedenen Dateiversionen einer Datei leichter analysieren können (diese werden häufig von Entwicklungsteams und in Code-Repositories wie GitHub verwendet).

INCLUDE-DATEIEN VERWENDEN

PHP

section_a/c02/include-and-require-files.php

```
<?php  
$stock = 25;  
  
if ($stock >= 10) {  
    $message = 'Good availability';  
}  
① if ($stock > 0 && $stock < 10) {  
    $message = 'Low stock';  
}  
if ($stock == 0) {  
    $message = 'Out of stock';  
}  
?>
```

② <?php require_once 'includes/header.php'; ?>

③ <h2>Chocolate</h2>
<p><?= \$message ?></p>

④ <?php include 'includes/footer.php'; ?>

ERGEBNIS



Diese Seite greift auf die links gezeigten Include-Dateien zurück.

1. Die Seite erzeugt zunächst eine Nachricht über den Lagerbestand und speichert diese in einer Variablen \$message. Wenn der Lagerbestand:

- größer oder gleich 10 ist, lautet die Nachricht Good availability.
- zwischen 1 und 9 ist, lautet die Nachricht Low stock.
- 0 ist, lautet sie Out of stock.

2. Zu Beginn des HTML-Teils der Seite wird die Header-Datei eingebunden (sie enthält den Code, der am Anfang jeder Seite steht).

Die Anweisung require_once besagt, dass die Datei nur einmal eingebunden werden soll. Sie wird dort platziert, wo der Header angezeigt werden soll, und der PHP-Interpreter behandelt sie so, als sei dieser Code kopiert und dort wieder eingefügt worden.

3. Es folgt der sichtbare Seiteninhalt, der die Lagerbestandsmeldung ausgibt.

4. Die Anweisung include weist den PHP-Interpreter an, den Code aus der Datei footer.php einzubinden.

Probieren Sie es: Fügen Sie in header.php einen neuen Link in die Navigation ein.

The Candy Store

[HOME](#) | [CANDY](#) | [ABOUT](#) | [CONTACT](#)

Welcome back, Ivy

LOLLIPOP DISCOUNTS

PACKS	PRICE
1 pack	\$1.92
2 packs	\$3.68
3 packs	\$5.28
4 packs	\$6.72
5 packs	\$8
6 packs	\$9.12
7 packs	\$10.08
8 packs	\$10.88
9 packs	\$11.52
10 packs	\$12

© 2021

BEISPIEL

Dieses Beispiel umfasst eine Begrüßung, gefolgt von den Rabatten, die beim Kauf mehrerer Packungen Süßigkeiten gewährt werden. Es nutzt dabei eine Reihe von Techniken, die in diesem Kapitel vorgestellt wurden.

- Der Name des Benutzers oder der Benutzerin wird in einer Variablen gespeichert.
- Eine bedingte Anweisung erzeugt eine Begrüßung.
- Mithilfe einer `for`-Schleife entsteht ein indiziertes Array, mit ermäßigten Preisen für den Kauf mehrerer Süßwarenpäckchen.
- Die Kopf- und Fußbereiche der Seite werden in Include-Dateien ausgelagert.
- Mithilfe einer `foreach`-Schleife werden die ermäßigten Preise aus dem Array angezeigt. Für ein Päckchen gibt es 4% Preisnachlass, für zwei Päckchen beträgt der Rabatt 8%, für drei Päckchen 12% und so weiter.
- Ein ternärer Operator gewährleistet, dass die Seite für eine einzelne Süßigkeitenpackung das Wort `pack` anzeigt und für mehrere Päckchen entsprechend die Mehrzahl `packs`.

BEISPIEL

Zunächst werden in der Datei die Werte erstellt und in Variablen gespeichert, die auf der Seite angezeigt werden sollen.

1. Die Variable `$name` enthält den Namen des Benutzers oder der Benutzerin.
2. Die Variable `$greeting` wird initialisiert und enthält zunächst die Nachricht Hello, die jedem angezeigt werden kann.
3. Eine if-Anweisung prüft, ob die Variable `$name` einen Wert hat.
4. Trifft dies zu, wird der Wert in `$greeting` mit einer personalisierten Begrüßung überschrieben.
5. In `$product` wird ein Produktnamen gespeichert.
6. Der Preis für ein Päckchen wird in `$cost` gespeichert.
7. Eine for-Schleife erzeugt ein Array, in dem die Preise für mehrere Süßigkeitenpäckchen gespeichert werden. Der Zähler steht für die Anzahl der Päckchen. Innerhalb der Klammern:
 - Der Zähler wird auf 1 gesetzt (dies entspricht einem Paket Süßigkeiten)
 - Die Bedingung überprüft, ob der Zähler kleiner oder gleich 10 ist (das wären 10 Päckchen Süßigkeiten)
 - Der Zähler erhöht sich bei jedem DurchlaufIn der Schleife:
8. Die Variable `$subtotal` enthält das Produkt aus dem Preis einer einzelnen Packung Süßigkeiten und dem Zähler (der die aktuelle Anzahl an Packungen darstellt).
9. Die Variable `$discount` enthält den Rabatt beim Kauf dieser Packungsanzahl. Dieser berechnet sich aus dem Preis geteilt durch 100 und multipliziert mit dem Zähler und dem Wert 4.
10. Die Variable `$totals` enthält ein Array; der Schlüssel ist der aktuelle Zählerwert (die Packungsanzahl), und der Wert ist der Preis für diese Packungsanzahl abzüglich des Rabatts.
11. Die for-Schleife wird geschlossen.

Dieser Seitenteil erzeugt den HTML-Code, der an den Browser zurückgesandt wird:

12. Der Seiten-Header wird mit dem Schlüsselwort `require` eingebunden, da er benötigt wird, um den Rest der Seite korrekt anzuzeigen (die Header-Datei haben wir auf Seite 96 vorgestellt).
 13. Die Begrüßung wird auf der Seite ausgegeben.
 14. Der Produktnamen wird auf der Seite ausgegeben.
 15. Eine HTML-Tabelle wird angelegt, und die erste Zeile wird mit Spaltenüberschriften versehen.
 16. Eine foreach-Schleife dient zur Anzeige der Daten, die im in den Schritten 7–11 erstellten Array gespeichert wurden.

Für jedes Element in diesem Array wird eine neue Datenzeile in die Tabelle eingefügt. In den Klammern geschieht Folgendes:

 - Das zu verwendende Array wird in der Variablen `$totals` gespeichert
 - Der Schlüssel wird in der Variablen `$quantity` gespeichert
 - Der Wert wird in der Variablen `$price` gespeichert
 17. Der Schlüssel für dieses Array-Element wird ausgegeben (dies ist die Anzahl der Süßigkeitspackungen).
 18. Die Zeichenkette `pack` wird ausgegeben. Die Bedingung eines ternären Operators überprüft dann, ob der Wert in `$quantity` gleich 1 ist. Wenn ja, wird nichts angehängt. Weicht der Wert von 1 ab, wird ein s angehängt (sodass auf der Seite `packs` statt des Singularen `pack` steht).
 19. Der (rabattierte) Preis für diese Menge an Süßigkeiten wird ausgegeben.
 20. Die foreach-Schleife wird geschlossen.
 21. Der Fußbereich der Seite wird mit dem Schlüsselwort `include` eingebunden (die Footer-Datei haben wir auf Seite 96 vorgestellt).
- Probieren Sie es:** Ändern Sie in Schritt 6 den Wert in `$cost` auf 10.
- Überarbeiten Sie in Schritt 7 die Schleife so, dass sie 20-mal ausgeführt wird und die Preise für bis zu 20 Päckchen Süßigkeiten ausgibt.

```
<?php  
① $name = 'Ivy';                                // Benutzername speichern  
  
② $greeting = 'Hello';                          // Startwert für Begrüßung definieren  
③ if ($name) {                                 // Wenn $name einen Wert enthält  
④     $greeting = 'Welcome back, ' . $name;    // personalisierte Begrüßung erzeugen  
    }  
  
⑤ $product = 'Lollipop';                        // Produktname  
⑥ $cost    = 2;                                  // Preis für einzelnes Päckchen  
  
⑦ for ($i = 1; $i <= 10; $i++) {  
    $subtotal  = $cost * $i;                      // Gesamtpreis für diese Menge  
    $discount  = ($subtotal / 100) * ($i * 4);    // Rabatt für diese Menge  
    $totals[$i] = $subtotal - $discount;          // Rabattpreis in indiziertes Array  
⑪ }  
?>  
  
⑫ <?php require 'includes/header.php'; ?>  
  
⑬ <p><?= $greeting ?></p>  
⑭ <h2><?= $product ?> Discounts</h2>  
⑮ [  <table>  
    <tr>  
        <th>Packs</th>  
        <th>Price</th>  
    </tr>  
    <?php foreach ($totals as $quantity => $price) { ?>  
    <tr>  
        <td>  
            <?= $quantity ?>  
            pack<?= ($quantity === 1) ? '' : 's'; ?>  
        </td>  
        <td>  
            $<?= $price ?>  
        </td>  
    </tr>  
    <?php } ?>  
  </table>  
⑯ <?php include 'includes/footer.php' ?>
```

ZUSAMMENFASSUNG

KONTROLLSTRUKTUREN

- **Geschweifte Klammern fassen eine Gruppe zusammenhängender Anweisungen zu einem Code-Block zusammen.**
- **Bedingte Anweisungen entscheiden anhand einer Bedingung, ob ein Code-Block ausgeführt werden soll oder nicht.**
- **Bedingungen liefern entweder den Wert true oder false.**
- Es gibt fünf Arten von bedingten Anweisungen:
`if`, `if... else`, `if... elseif`, `switch` und `match`.
- **Mit Schleifen können Sie denselben Code-Block mehrmals wiederholen, solange eine Bedingung erfüllt bleibt.**
- **Es gibt vier Arten von Schleifen: `while`, `do... while`, `for` und `foreach`.**
- **Include-Dateien enthalten Code, der auf mehreren Seiten verwendet werden soll, sodass Sie ihn nicht mehrfach ausschreiben müssen.**

3

FUNKTIONEN

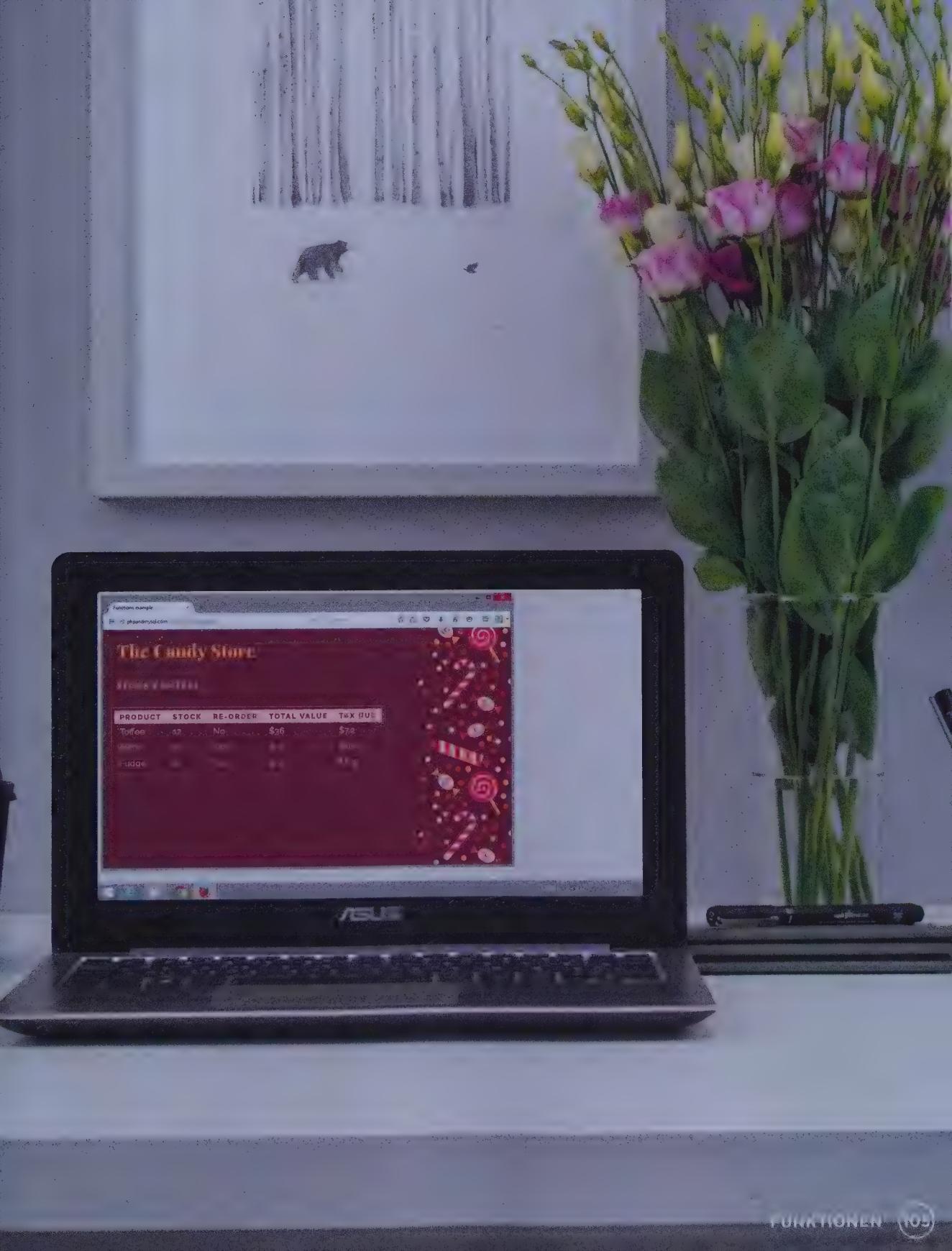
Oft erfüllt eine einzige Webseite zahlreiche Aufgaben. Funktionen organisieren Ihren Code, indem sie Anweisungen zusammenfassen, die für die Ausführung einer Aufgabe erforderlich sind.

Eine PHP-Seite kann Hunderte von Code-Zeilen enthalten und viele verschiedene Aufgaben erfüllen. Daher ist es wichtig, dass der Code sorgfältig strukturiert ist, damit Sie (und andere) leicht verstehen können, was er bewirkt.

Im letzten Kapitel haben Sie gesehen, dass Sie Ihren Code mit Code-Blöcken organisieren können, indem Sie eine Abfolge von zusammenhängenden Anweisungen in geschweifte Klammern setzen. Diese teilen dem PHP-Interpreter mit, wo der Satz von zusammenhängenden Anweisungen beginnt und endet. Das bedeutet, dass ein Code-Block übersprungen (mit bedingten Anweisungen) oder wiederholt werden kann (mit Schleifen).

Eine Funktion fasst alle Anweisungen zusammen, die zur Ausführung einer einzelnen Aufgabe innerhalb eines Code-Blocks erforderlich sind. Sie gibt dem Code-Block auch einen Funktionsnamen, der die auszuführende Aufgabe beschreibt (sodass Sie den Code für eine bestimmte Aufgabe leichter finden können). Die öffnende geschweifte Klammer teilt dem PHP-Interpreter mit, wo die Anweisungen zur Ausführung der Aufgabe beginnen, und die zugehörige schließende geschweifte Klammer zeigt ihm, wo sie enden.

Wenn der PHP-Interpreter auf eine Funktion stößt, führt er den Code nicht sofort aus. Er wartet, bis eine andere Anweisung in der PHP-Seite die Funktion mit ihrem Namen aufruft; erst dann führt er die Anweisungen im Code-Block aus. Sie können den PHP-Interpreter auch anweisen, die Funktion mehrfach zu verwenden, damit Sie nicht dieselben Code-Zeilen wiederholen müssen, wenn Sie die Aufgabe mehr als einmal ausführen müssen.



FUNKTIONEN VERWENDEN

Jede Aufgabe, die eine PHP-Seite erfüllt, kann zahlreiche PHP-Anweisungen erfordern. Die Anweisungen, die zur Ausführung einer einzelnen Aufgabe erforderlich sind, können in einer Funktion gespeichert und bei Bedarf aufgerufen werden.

EINE FUNKTION DEFINIEREN UND AUFRUFEN

Siehe Seiten 108 bis 111

Eine Funktion wird erstellt, indem man ihr einen beschreibenden Namen gibt. Darauf folgen die Anweisungen, die zur Ausführung der Aufgabe in einem Code-Block erforderlich sind. Man spricht von einer **Funktionsdefinition**.

Wenn die Seite die Aufgabe ausführen soll, wird über den Funktionsnamen dem PHP-Interpreter mitgeteilt, dass er die Anweisungen in diesem Code-Block ausführen soll. Man bezeichnet dies als **Funktionsaufruf**.

DATEN AUS FUNKTIONEN ABRUFEN

Siehe Seiten 112 bis 113

Wenn eine Funktion ihre Aufgabe erfüllt hat, **gibt** sie in der Regel einen Wert mit dem Ergebnis der von ihr ausgeführten Aufgabe **zurück**. Zwei Beispiele:

- Soll eine Nutzerin mithilfe der Funktion auf der Website angemeldet werden, könnte die Funktion `true` zurückgeben, wenn die Nutzerin sich erfolgreich anmeldet, anderenfalls `false`.
- Wenn eine Funktion zur Berechnung des Gesamtpreises einer Bestellung verwendet wird, gibt sie diese Summe zurück.

FUNKTIONEN DEFINIEREN, DIE DATEN BENÖTIGEN

Siehe Seiten 114 bis 116

Funktionen benötigen oft Informationen, um ihre Aufgabe erfüllen zu können.

Soll ein Benutzer an der Website angemeldet werden, benötigt die Funktion möglicherweise zwei Daten: die E-Mail-Adresse des Benutzers und sein Passwort.

Parameter sind wie Variablennamen, die für die einzelnen Daten stehen, die die Funktion zur Erfüllung ihrer Aufgabe benötigt. Die tatsächlichen beim Aufruf der Funktion verwendeten Werte werden als **Argumente** bezeichnet.

Die in diesem Kapitel erläuterten Funktionen erfüllen sehr einfache Aufgaben, damit Sie lernen können, wie Funktionen erstellt und warum sie verwendet werden. In späteren Kapiteln beschäftigen wir uns mit Funktionen für komplexere Aufgaben.

WIE VARIABLEN MIT FUNKTIONEN ZUSAMMENWIRKEN

Siehe Seiten 118 bis 121

Der Code innerhalb einer Funktion kann nicht auf außerhalb der Funktion deklarierte Variablen zugreifen; daher müssen alle von der Funktion benötigten Daten über einen Parameter an die Funktion übergeben werden.

Ebenso kann der Code außerhalb der Funktion nicht auf Variablen zugreifen, die in einer Funktion deklariert wurden. Aus diesem Grund ist eine Funktion oft so konstruiert, dass sie einen Wert an den Code zurückgibt, der sie aufgerufen hat (nachdem die Funktion ausgeführt wurde).

DATENTYPEN ANGEBEN

Siehe Seiten 124 bis 127

Typdeklarationen teilen dem PHP-Interpreter den Datentyp mit, den sie erwarten:

- an eine Funktion (als Argument) übergeben
- von einer Funktion zurückgeben

Durch Typendeklarationen stellen Sie sicher, dass die Funktion Daten erhält, die sie zur Ausführung ihrer Aufgabe verwenden kann. Sie helfen auch, Probleme im Code aufzuspüren, wenn er nicht wie erwartet funktioniert.

OPTIONALE UND STANDARDWERTE

Siehe Seiten 130 bis 131

Wenn Sie eine Funktion erstellen, können Sie angeben, dass einer oder mehrere der Parameter (Informationen, die zur Erfüllung der Aufgabe benötigt werden) optional sind und nicht angegeben werden müssen.

Wenn Sie bestimmen, dass ein Parameter optional ist, müssen Sie einen Standardwert für den Parameter angeben. Dies ist ein Wert, der verwendet werden soll, wenn die Funktion ohne einen Wert für diesen Parameter aufgerufen wird.

EINE FUNKTION DEFINIEREN UND AUFRUFEN

Eine **Funktionsdefinition** speichert die Anweisungen für eine Aufgabe in geschweiften Klammern, um einen Code-Block zu bilden, und gibt ihnen einen Namen, der die Aufgabe beschreibt. Die Funktion wird **aufgerufen**, sobald die Aufgabe ausgeführt werden muss.

Um eine Funktion zu definieren (oder zu erstellen), verwenden Sie:

- das Schlüsselwort `function` (dies zeigt an, dass Sie eine neue Funktion definieren)
- einen Namen, der die Aufgabe der Funktion beschreibt, gefolgt von einem Klammerpaar
- geschweifte Klammern für den Code, der die Aufgabe ausführt

Nach der schließenden geschweiften Klammer folgt kein Semikolon.

Die folgende Funktion enthält zwei Anweisungen:

- Die erste speichert das aktuelle Jahr in der Variablen `$year` (wie das funktioniert, sehen Sie in Kapitel 8).
- Die zweite schreibt einen Copyright-Hinweis auf die Seite, wobei der in der Variablen `$year` gespeicherte Wert verwendet wird.

Wenn Sie eine Funktion definieren, wird der Code nicht ausgeführt. Er wird lediglich in der Funktionsdefinition zur späteren Verwendung gespeichert.

Das Diagramm zeigt den Quellcode einer PHP-Funktion. Die Wörter 'SCHLÜSSELWORT' und 'NAME' sind über dem Schlüsselwort 'function' und den Namen 'write_copyright_notice()' angeordnet. Unter dem Wort 'ÖFFNENDE GESCHWEIFTE KLAMMER' steht ein geschweiftes Klammerzeichen '{'. Unter dem Wort 'GESCHWEIFTE KLAMMER' steht ein geschweiftes Klammerzeichen '}'. Die Zeile '\$year = date('Y');' ist als 'SCHLIESSENDE —' beschriftet.

```
function write_copyright_notice()  
{  
    $year = date('Y');  
    echo '&copy; ' . $year;  
}
```

Wenn Sie eine Aufgabe ausführen müssen, die in einer Funktion definiert ist, geben Sie den Funktionsnamen gefolgt von Klammern an. Dies vermittelt dem PHP-Interpreter, dass Sie die Anweisungen in der Funktion ausführen wollen.

Sie können dieselbe Funktion innerhalb einer PHP-Datei so oft wie gewünscht aufrufen. Sobald eine Funktion ihre Aufgabe erfüllt hat, fährt der PHP-Interpreter mit der Codezeile fort, die auf die Zeile folgt, in der die Funktion aufgerufen wurde.

Das Diagramm zeigt den Aufruf einer Funktion. Das Wort 'NAME' ist über dem Funktionsnamen 'write_copyright_notice()' angeordnet.

```
write_copyright_notice();
```

GRUNDLEGENDE FUNKTIONEN

PHP

section_a/c03/basic-functions.php

```
<?php  
① function write_logo()  
{  
②     echo '';  
  
③ function write_copyright_notice()  
{  
④     $year = date('Y');  
⑤     echo '&copy; ' . $year;  
}  
?> ...  
<header>  
    <h1><?php write_logo(); ?> The Candy Store</h1>  
</header>  
<article>  
    <h2>Welcome to the Candy Store</h2>  
</article>  
<footer>  
    <?php write_logo(); ?>  
    <?php write_copyright_notice(); ?>  
</footer>
```

ERGEBNIS



Diese Seite verwendet zwei Funktionen.

- Die erste zeigt ein Logo an.
 - Die zweite erstellt einen Copyright-Hinweis.
- Die Funktion `write_logo()` wird definiert.
 - In den geschweiften Klammern wird in einer einzigen Anweisung ein Logo angezeigt.
 - Die Funktion `write_copyright_notice()` wird definiert. Die geschweiften Klammern enthalten zwei Anweisungen.
 - Das aktuelle Jahr wird in der Variablen `$year` gespeichert (wie das funktioniert, erfahren Sie in Kapitel 8).
 - Das Copyright-Symbol © wird auf die Seite geschrieben, gefolgt von der Jahreszahl.
 - Die erste Funktion wird aufgerufen, um ein Logo oben auf der Seite einzufügen.
 - Dieselbe Funktion wird aufgerufen, um ein Logo in der Fußzeile einzufügen.
 - Die zweite Funktion wird aufgerufen, um den Copyright-Hinweis in der Fußzeile der Seite einzufügen.
- Probieren Sie es:** Schreiben Sie nach Schritt 5 innerhalb der Funktion den Firmennamen (The Candy Store) nach dem Copyright-Hinweis.

CODE WIRD NICHT IMMER SEQUENZIELL AUSGEFÜHRT

Eine Funktionsdefinition speichert die Anweisungen, die zur Ausführung einer Aufgabe erforderlich sind. Sie werden nur ausgeführt, wenn die Funktion aufgerufen wird. Das bedeutet, dass der Code nicht immer in derselben Reihenfolge ausgeführt wird, in der er geschrieben wurde.

Wenn man zum ersten Mal PHP-Code sieht, nimmt man oft an, dass die Anweisungen in der Reihenfolge ausgeführt werden, in der sie notiert wurden. In der Praxis kann der PHP-Interpreter die Anweisungen jedoch in einer ganz anderen Reihenfolge ausführen.

Oft sehen Sie Funktionen, die im oberen Bereich der PHP-Seite definiert sind. (Wenn auch Variablen am Anfang der Seite deklariert wurden, stehen diese normalerweise vor den Funktionsdefinitionen.)

Die Funktionen werden dann später im Code dort aufgerufen, wo die Aufgabe ausgeführt werden muss.

Wie Sie auf diesen beiden Seiten sehen, werden in einer Funktionsdefinition nur die Anweisungen in einem Code-Block gespeichert (und der Funktion ein Name gegeben, um ihre Funktionsweise zu beschreiben), auch wenn die Funktionen am Anfang der Seite geschrieben werden. Der PHP-Interpreter führt diesen Code erst aus, wenn die Funktion aufgerufen wird. Das kann bedeuten, dass die Anweisungen in einer ganz anderen Reihenfolge ausgeführt werden, als sie im Code erscheinen.

```
<?php  
① function write_logo()  
{  
②     echo '';  
}  
  
③ function write_copyright_notice()  
{  
④     $year = date('Y'); // Jahr abrufen und speichern  
⑤     echo '&copy; ' . $year; // Copyright-Hinweis  
}  
?  
<!DOCTYPE html>  
<html>  
    <header>  
        ⑥     <h1><?php write_logo(); ?> The Candy Store</h1>  
    </header>  
    <article>  
        <p>Welcome to The Candy Store</p>  
    </article>  
    <footer>  
        ⑦     <?php write_logo(); ?>  
        <?php write_copyright_notice(); ?>  
    </footer>  
</html>
```

Die linke Seite zeigt das Beispiel auf der vorhergehenden Seite. Darunter sehen Sie die Reihenfolge, in der die Anweisungen ausgeführt werden. Die erste Zeile, die der PHP-Interpreter tatsächlich ausführt, befindet sich in Schritt 6.

SCHRITT **DIE AUFGABEN DES INTERPRETERS**

- 6** Die erste Zeile, die ausgeführt wird, ist Schritt 6. Sie ruft die Funktion `write_logo()` auf.
- 1, 2** Der PHP-Interpreter geht zu Schritt 1, in dem die Funktion definiert wurde, und führt dann Schritt 2 aus.
- 6** Nachdem die Funktion ausgeführt wurde, kehrt der Interpreter zu der Codezeile zurück, in der die Funktion aufgerufen wurde.
- 7** Er geht nun zur nächsten PHP-Codezeile über. In Schritt 7 wird die Funktion `write_logo()` erneut aufgerufen.
- 1, 2** Der PHP-Interpreter geht zu Schritt 1, wo die Funktion definiert wurde, und führt dann Schritt 2 aus.
- 7** Nachdem die Funktion ausgeführt wurde, kehrt der Interpreter zu der Codezeile zurück, in der die Funktion aufgerufen wurde.
- 8** Er geht nun zur nächsten PHP-Codezeile über. Schritt 8 ruft die Funktion `write_copyright_notice()` auf.
- 3, 4, 5** Er fährt mit Schritt 3 fort, wo die Funktion deklariert wurde, und führt dann die Schritte 4–5 aus.
- 8** Sobald die Funktion beendet ist, kehrt sie zu der Zeile zurück, in der sie aufgerufen wurde.

In bestimmten Fällen wird eine Funktion aufgerufen, bevor sie definiert ist, aber es ist besser, Funktionen zu definieren, bevor man sie aufruft.

Wenn in mehreren Seiten dieselben Funktionen benötigt werden, können die Funktionsdefinitionen in einer Include-Datei gespeichert werden.

DATEN AUS FUNKTIONEN ABRUFEN

Mit Funktionen werden häufig neue Werte erzeugt. Diese können mit dem Schlüsselwort `return` an die Anweisung, die die Funktion aufgerufen hat, zurückgeschickt werden.

Funktionen schreiben selten Daten direkt auf die Seite (wie im vorherigen Beispiel). Häufiger erstellt eine Funktion einen neuen Wert und gibt ihn an die Anweisung zurück, die sie aufgerufen hat. Um einen Wert zurückzugeben, verwenden Sie das Schlüsselwort `return`, gefolgt von dem Wert, den Sie zurückgeben möchten.

```
function create_copyright_notice()
{
    $year      = date('Y');
    $message = '&copy; ' . $year;
    return $message;
}
```

SCHLÜSSELWORT RÜCKGABEWERT

Die folgende Funktion ähnelt den Funktionen auf den vier vorangegangenen Seiten, aber sie erstellt einen Copyright-Hinweis und speichert ihn in der Variablen `$message`. Anschließend gibt sie den in der Variablen `$message` gespeicherten Wert zurück (anstatt den HTML-Code direkt in die Seite zu schreiben).

Wenn eine Funktion einen Wert zurückgibt und Sie diese Daten auf der Seite anzeigen möchten, verwenden Sie den Befehl `echo` (oder die Kurzform dafür) und rufen dann die Funktion auf.

Es gilt als bessere Praxis, einen Wert aus einer Funktion zurückzugeben und diesen dann in die Seite zu schreiben, anstatt einen `echo`-Befehl innerhalb der Funktion zu verwenden.

```
<?= create_copyright_notice() ?>
```

Sie können den von einer Funktion zurückgegebenen Wert auch in einer Variablen speichern.

Verwenden Sie dazu einen Variablenamen, gefolgt von einem Zuweisungsoperator, und rufen Sie die Funktion auf.

```
$copyright_notice = create_copyright_notice();
```

FUNKTIONEN, DIE EINEN WERT ZURÜCKGEBEN

PHP

section_a/c03/functions-with-return-values.php

```
<?php  
① function create_logo()  
{  
②     return '';  
}  
  
③ function create_copyright_notice()  
{  
④     $year      = date('Y');  
⑤     $message = '&copy; ' . $year;  
⑥     return $message;  
}  
?> ...  
<header>  
⑦     <h1><?= create_logo() ?>The Candy Store</h1>  
</header>  
<article>  
    <h2>Welcome to The Candy Store</h2>  
</article>  
<footer>  
    <?= create_logo() ?>  
    <?= create_copyright_notice() ?>  
</footer>
```

ERGEBNIS



In diesem Beispiel werden die Funktionen so angepasst, dass sie Werte zurückgeben.

1. `create_logo()` wird definiert.
2. Das Schlüsselwort `return` wird verwendet, gefolgt vom HTML-Code für die Bilderstellung.
3. `create_copyright_notice()` wird definiert.

In den geschweiften Klammern befinden sich drei Anweisungen, die:

4. das aktuelle Jahr abfragen und in der Variablen `$year` speichern
5. die Variable `$message` erstellen und darin das Copyright-Symbol © speichern, gefolgt von der aktuellen Jahreszahl
6. den in der Variablen `$message` gespeicherten Wert zurückgeben
7. Die erste Funktion wird aufgerufen, und der zurückgegebene Wert wird mit der Kurzform für `echo` in die Seite geschrieben.
8. Die erste Funktion wird erneut aufgerufen, um das Logo zu wiederholen.
9. Die zweite Funktion wird aufgerufen, und der zurückgegebene Wert wird in die Seite geschrieben, wobei die Kurzform für `echo` verwendet wird.

Probieren Sie es: Fügen Sie in Schritt 5 den Firmennamen in die Variable `$message` ein.

FUNKTIONEN, DIE DATEN BENÖTIGEN, DEFINIEREN

Parameter sind wie Variablennamen für die Werte, die eine Funktion zur Erfüllung ihrer Aufgabe benötigt. Die Werte, für die der Parameter steht, können sich bei jedem Funktionsaufruf ändern.

Geben Sie bei der Definition einer Funktion an, ob sie Daten benötigt, um ihre Aufgabe zu erfüllen:

- Listen Sie die Informationen auf, die sie benötigt.
- Geben Sie jeder dieser Informationen einen Variablenamen (beginnend mit \$), der die Art der Daten beschreibt.
- Setzen Sie diese Namen in die Klammern, die hinter dem Funktionsnamen stehen.
- Trennen Sie die einzelnen Namen durch Kommas.
- Diese werden als **Parameter** der Funktion bezeichnet.

Parameter verhalten sich wie Variablen, können aber nur von den Anweisungen in den geschweiften Klammern der Funktionsdefinition verwendet werden. Code außerhalb der Funktionsdefinition kann nicht auf sie zugreifen.

Die nachstehende Funktion `calculate_cost()` berechnet die Gesamtsumme, wenn die Benutzer einen oder mehrere Artikel desselben Typs kaufen. Um diese Aufgabe zu erfüllen, benötigt sie zwei Parameter:

- `$price` steht für den Preis eines Artikels.
- `$quantity` steht für die Menge dieses Artikels.

Innerhalb dieser Funktion verhalten sich `$price` und `$quantity` wie Variablen. Sie stellen die Werte dar, die beim Funktionsaufruf an die Funktion übergeben werden.

Der Code innerhalb der Funktionsdefinition berechnet die Gesamtpreis, indem er den Preis mit der Menge multipliziert.

Dieser Wert wird dann mit dem Schlüsselwort `return` an den Code zurückgesendet, der die Funktion aufgerufen hat.

```
function calculate_cost($price, $quantity)
{
    return $price * $quantity;
}
```

PARAMETER
PARAMETERNAMEN WERDEN INNERHALB DER FUNKTION
WIE VARIABLEN VERWENDET

Wenn der PHP-Interpreter die schließende geschweifte Klammer erreicht, verwirft er alle in der Funktion gespeicherten Werte. Dies ist wichtig, weil eine Seite eine Funktion mehrmals aufrufen kann, wobei jedes Mal andere Werte verwendet werden.

Hinweis: In PHP 8 kann die Funktionsdefinition ein Komma nach dem letzten Parameternamen aufweisen (nicht nur zwischen den Parametern), wodurch der Code einheitlicher wird. Dies würde in früheren PHP-Versionen einen Fehler verursachen.

FUNKTIONEN MIT PARAMETERN UND WERTEN

Beim Aufruf einer Funktion mit Parametern wird der Wert für jeden Parameter in Klammern hinter dem Funktionsnamen angegeben. Die Werte, die beim Aufruf einer Funktion verwendet werden, werden als **Argumente** bezeichnet.

ARGUMENTE ALS WERTE

Wenn die Funktion `calculate_cost()` aufgerufen wird, werden ihr die zu verwendenden Werte übergeben.

Die Werte werden in der gleichen Reihenfolge übergeben, in der die Parameter in der Funktionsdefinition angegeben wurden.

```
$total = calculate_cost(3, 5);
```

Die Zahl 3 wird für den Preis des Artikels verwendet, und die Zahl 5 wird für die gekaufte Menge verwendet, demnach gibt `calculate_cost()` die Zahl 15 zurück, und dieser Wert wird in der Variablen `$total` gespeichert.

ARGUMENTE ALS VARIABLEN

Diesmal werden beim Aufruf von `calculate_cost()` nicht Werte, sondern Variablennamen verwendet:

- `$cost` steht für den Preis des Artikels.
- `$units` steht für die gekaufte Menge.

Wenn Variablennamen als Argumente verwendet werden, müssen die Variablennamen nicht mit den Parameternamen übereinstimmen.

```
$cost = 4;  
$units = 6;  
$total = calculate_cost($cost, $units);
```

Wenn die unten stehende Funktion aufgerufen wird, sendet der PHP-Interpreter die in den Variablen `$cost` und `$units` gespeicherten Werte an die Funktion.

Innerhalb der Funktion werden diese Werte durch die Parameternamen `$price` und `$quantity` dargestellt (die Namen wurden in Klammern in der ersten Zeile der Funktionsdefinition angegeben).

PARAMETER VS. ARGUMENTE

Die Begriffe Parameter und Argumente werden oft synonym verwendet, aber es gibt einen feinen Unterschied. Wenn die Funktion definiert ist, sehen Sie auf der linken Seite die Namen `$price` und `$quantity`.

Innerhalb der geschweiften Klammern der Funktion verhalten sich diese Wörter wie Variablen. Diese Namen sind die **Parameter**.

Wird die Funktion auf dieser Seite aufgerufen, gibt der Code entweder Zahlen an, die für die Berechnung verwendet werden, oder Variablen, die Zahlen enthalten. Diese Werte, die dem Code übergeben werden (die Informationen, die für die Berechnung dieser speziellen Süßigkeitensorte benötigt werden), werden als **Argumente** bezeichnet.

EINE FUNKTION MIT PARAMETERN

1. calculate_total() wird oben auf der Seite definiert. Sie berechnet die Gesamtsumme, wenn ein oder mehrere gleichartige Artikel gekauft werden, und fügt dann 20 % Steuer hinzu. Sie benötigt zwei Daten und hat daher zwei Parameter:

- \$price steht für den Preis eines einzelnen Artikels.
- \$quantity für die Anzahl der Einheiten des gekauften Artikels.

2. In der Funktionsdefinition speichert eine Variable \$cost die Preise für die erforderliche Anzahl Einheiten. Diese werden durch Multiplikation des in \$price gespeicherten Werts mit dem Wert in \$quantity berechnet.

3. Nun wird die für diese Artikel anfallende Umsatzsteuer in der Variablen \$tax gespeichert. Sie wird berechnet, indem der Wert in \$cost (erstellt in Schritt 2) mit 0,2 multipliziert wird.

4. Für die Gesamtsumme werden die Werte in \$cost und \$tax addiert.

5. Die Summe wird von der Funktion an den Code zurückgegeben, der sie aufgerufen hat.

6. Die Funktion wird dreimal aufgerufen. Jedes Mal werden andere Preise und Mengen verwendet, und der von der Funktion zurückgegebene Wert wird in die Seite geschrieben.

section_a/c03/function-with-parameters.php

PHP

```
<?php  
① function calculate_total($price, $quantity)  
 {  
 ②   $cost  = $price * $quantity;  
 ③   $tax   = $cost * (20 / 100);  
 ④   $total = $cost + $tax;  
 ⑤   return $total;  
 }  
 ?> ...  
<h1>The Candy Store</h1>  
⑥ <p>Mints: $<?= calculate_total(2, 5) ?></p>  
<p>Toffee: $<?= calculate_total(3, 5) ?></p>  
<p>Fudge: $<?= calculate_total(5, 4) ?></p>
```

ERGEBNIS



Probieren Sie es: Rufen Sie in Schritt 6 die Funktion erneut auf, um den Preis von 4 Packungen Kaugummi zu je 1,50 \$ anzuzeigen.

FUNKTIONEN BENENNEN

Der Name einer Funktion sollte ihren Zweck klar beschreiben. Er besteht oft aus einem beschreibenden Begriff und der Informationskategorie, mit der die Funktion arbeitet oder die sie zurückgibt.

Die Regeln für Funktionsnamen entsprechen denen für Variablen. Sie sollten mit einem Buchstaben beginnen, gefolgt von einer beliebigen Kombination aus Buchstaben, Zahlen oder Unterstrichen.

Auf ein und derselben PHP-Seite kann es nur eine Funktionen mit demselben Namen geben.

In diesem Buch werden alle Funktionsnamen in Kleinbuchstaben geschrieben.

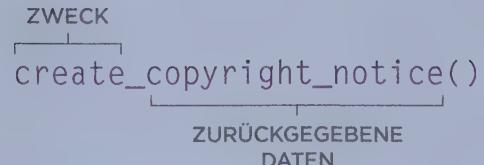
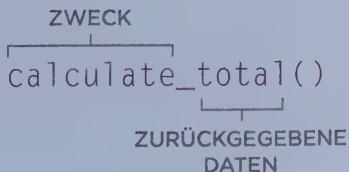
Wenn der Funktionsname mehrere Wörter erfordert, werden diese durch einen Unterstrich getrennt. (Sie werden sehen, dass Funktionen unterschiedliche Benennungsregeln verwenden; wichtig ist, dass Sie innerhalb eines Projekts eine einheitliche Benennungsstrategie nutzen).

Um einen Namen zu erstellen, der die von der Funktion ausgeführte Aufgabe beschreibt, gehen Sie so vor:

- Geben Sie an, was die Funktion tut (z.B. berechnen, abrufen oder aktualisieren)
- gefolgt von der Art der Information, die sie zurückgibt oder verarbeitet (z.B. Datum, Summe oder Nachricht)
- gefolgt von dem Informationstyp den sie zurückgibt oder verarbeitet (z.B. Datum, Gesamtsumme oder Nachricht)

Im Folgenden finden Sie zwei Beispiele für beschreibende Funktionsnamen, die Sie bereits in diesem Kapitel kennengelernt haben:

- `calculate_total()` berechnet den Gesamtpreis der zu verkaufenden Artikel.
- `create_copyright_notice()` erstellt einen Copyright-Hinweis.



GÜLTIGKEITSBEREICH

Wenn eine Funktion aufgerufen wird, wird der Code in seinem eigenen Gültigkeitsbereich ausgeführt; auf Werte, die in Variablen außerhalb der Funktion gespeichert sind, kann er nicht zugreifen oder diese aktualisieren.

Der Code in einer Funktion wird unabhängig vom Rest der Seite ausgeführt:

- Alle Informationen, die eine Funktion benötigt, um ihre Aufgabe zu erfüllen, müssen ihr über Parameter übergeben werden. Diese verhalten sich wie Variablen innerhalb der Funktion.
- Wenn die Funktion aufgerufen wird, können die Anweisungen in der Funktion Variablen erstellen und Ihnen Werte zuweisen.
- Die Funktion kann dann einen Wert an den Code zurückgeben, der die Funktion aufgerufen hat.
- Nachdem die Funktion ausgeführt wurde, werden alle in der Funktion erstellten Parameter und Variablen gelöscht.

Da der Code in der Funktion getrennt vom Rest der Seite ausgeführt wird:

- kann die Funktion nicht auf Variablen außerhalb der Funktion zugreifen oder diese aktualisieren (deshalb werden die Informationen als Parameter übergeben).
- kann der nachfolgende Code nicht auf die innerhalb der Funktion erstellten Variablen zugreifen, da diese entfernt werden, sobald die Funktion ihre Aufgabe erfüllt hat.

Jedes Mal, wenn eine Funktion aufgerufen wird, führt der PHP-Interpreter den Code für diese Funktion im **lokalen Gültigkeitsbereich** aus. Der Code außerhalb der Funktion im Hauptteil der Seite befindet sich im **globalen Gültigkeitsbereich**.

Der Ort, an dem eine Variable deklariert wird, d. h. im lokalen oder globalen Gültigkeitsbereich, bestimmt, ob andere Codes auf sie zugreifen können oder nicht.

Im folgenden Code gibt es zwei Variablen, die beide den Namen `$tax` tragen; jede hat einen anderen Gültigkeitsbereich:

- A:** Die erste Variable `$tax` wird im globalen Gültigkeitsbereich erstellt. (Sie liegt außerhalb der Funktion.)
B: Die zweite Variable mit dem Namen `$tax` wird innerhalb der Funktionsdefinition, im lokalen Gültigkeitsbereich, erstellt.

```
<?php  
① $tax = 20;  
function calculate_total($price, $quantity)  
{  
    $cost = $price * $quantity;  
    ② $tax = $cost * (20 / 100);  
    $total = $cost + $tax;  
    return $total;  
}  
?>
```

GÜLTIGKEITS-BEREICH: GLOBAL LOKAL

Im Idealfall sollten zwei Variablen im selben Skript nicht denselben Namen haben, aber dieses Beispiel zeigt, dass die Variablen völlig unabhängig voneinander betrachtet werden.

DEMONSTRATION DES GÜLTIGKEITSBEREICHES

PHP

section_a/c03/global-and-local-scope.php

```
<?php  
① $tax = '20';  
  
② function calculate_total($price, $quantity)  
{  
③     $cost = $price * $quantity;  
④     $tax = $cost * (20 / 100);  
⑤     $total = $cost + $tax;  
⑥     return $total;  
}  
?  
<h1>The Candy Store</h1>  
⑦ <p>Mints: $<?= calculate_total(2, 5) ?></p>  
<p>Toffee: $<?= calculate_total(3, 5) ?></p>  
<p>Fudge: $<?= calculate_total(5, 4) ?></p>  
⑧ <p>Prices include tax at: <?= $tax ?>%</p>
```

ERGEBNIS



Probieren Sie aus: Ändern Sie in Schritt 1 den Steuersatz auf 25. Dadurch ändert sich der unten auf der Seite angezeigte Steuersatz, aber nicht die in Schritt 7 ausgegebenen Summen.

Die Gesamtsummen in Schritt 7 bleiben gleich, weil in Schritt 4 der Funktion ein Steuersatz von 20 % verwendet wird. Dies zeigt, wie die beiden Variablen \$tax unabhängig voneinander funktionieren.

1. Die Variable \$tax wird im globalen Gültigkeitsbereich deklariert, sodass sie von jedem Code außerhalb der Funktion verwendet werden kann.

2. Die Funktion calculate_total() wird definiert. Sie benötigt den Preis und die Menge eines Artikels.

Die in dieser Funktion erstellten Variablen befinden sich im lokalen Gültigkeitsbereich.

3. Die Preise werden berechnet, indem der Preis des Artikels mit der benötigten Menge multipliziert wird. Das Ergebnis wird in der Variable \$cost gespeichert.

4. Die fällige Steuer wird durch Multiplikation von \$cost mit dem Steuersatz (20 geteilt durch 100) berechnet.

Das Ergebnis wird in der Variablen \$tax gespeichert. Hinweis: Dies überschreibt nicht den Wert, der in der in Schritt 1 erstellten Variablen \$tax gespeichert ist.

5. Um die Gesamtsumme zu ermitteln, wird der Wert in \$cost zu dem Wert addiert, der in Schritt 4 in \$tax gespeichert wurde.

6. Die Gesamtsumme wird zurückgegeben. Wenn die Funktion ausgeführt wurde, löscht der PHP-Interpreter alle Parameter und die in der Funktion erstellten Variablen.

7. Die Funktion wird dreimal mit jeweils neuen Werten aufgerufen.

8. Der in \$tax gespeicherte Wert (im globalen Gültigkeitsbereich in Schritt 1) wird angezeigt.

GLOBALE UND STATISCHE VARIABLEN

In wenigen Fällen kann der Code in einer Funktion auf eine globale Variable zugreifen oder diese aktualisieren und einen Wert speichern, der in einer Variablen in der Funktion gespeichert wurde, nachdem diese beendet wurde.

ZUGRIFF AUF ODER AKTUALISIERUNG VON GLOBALEN VARIABLEN INNERHALB EINER FUNKTION

Der Code innerhalb einer Funktion kann auf einen Wert zugreifen, der in einer im globalen Gültigkeitsbereich deklarierten Variablen gespeichert ist, oder diesen aktualisieren, wenn dem PHP-Interpreter erlaubt wird, auf diese Variable zuzugreifen.

Fügen Sie am Anfang des Code-Blocks der Funktion (bevor die Variable verwendet wird) das Schlüsselwort `global` ein, gefolgt vom Namen der Variablen. Dadurch kann der Code in der Funktion auf ihren Wert zugreifen oder ihn aktualisieren.

Es gilt als bewährte Praxis, Werte mithilfe von Parametern an eine Funktion zu übergeben, aber diese Möglichkeit wird hier erwähnt, weil Sie möglicherweise auf Code stoßen werden, der mit dieser Technik auf eine globale Variable zugreift.

```
global $cost;
```

SCHLÜSSELWORT VARIABLE

EINEN WERT IN EINER FUNKTION NACH DEREN AUSFÜHRUNG BEIBEHALTEN

Nach Beendigung einer Funktion werden normalerweise alle lokalen Variablen, die innerhalb der Funktion erstellt wurden, entfernt.

Dem PHP-Interpreter kann mitgeteilt werden, dass er sich einen Wert, der in einer in einer Funktion erstellten Variablen gespeichert ist, merken soll, wenn diese als **statische Variable** erstellt wurde.

Um eine statische Variable zu erstellen, verwenden Sie

- das Schlüsselwort `static`,
- gefolgt von einem Variablenamen,
- dann einen Anfangswert, den sie beim ersten Aufruf der Funktion enthalten soll.

Wenn die Funktion beendet ist, werden diese Variable und der in ihr gespeicherte Wert nicht gelöscht (sie sind aber nur für den Code innerhalb der Funktion verfügbar).

```
static $quantity = 10;
```

SCHLÜSSELWORT VARIABLE INITIALWERT

ZUGRIFF AUF VARIABLEN AUSSERHALB EINER FUNKTION

PHP

section_a/c03/global-and-static-variables.php

```
<?php
① $tax_rate = 0.2;

② function calculate_running_total($price, $quantity)
{
    ③ global $tax_rate;
    ④ static $running_total = 0;
    ⑤ $total = $price * $quantity;
    ⑥ $tax   = $total * $tax_rate;
    ⑦ $running_total = $running_total + $total + $tax;
    ⑧ return $running_total;
}
?> ...
<h1>The Candy Store</h1>
<table>
    <tr><th>Item</th><th>Price</th><th>Qty</th>
        <th>Running total</th></tr>
    <tr><td>Mints:</td><td>$2</td><td>5</td>
        <td>$<?= calculate_running_total(2, 5); ?></td></tr>
    <tr><td>Toffee:</td><td>$3</td><td>5</td>
        <td>$<?= calculate_running_total(3, 5); ?></td></tr>
    <tr><td>Fudge:</td><td>$5</td><td>4</td>
        <td>$<?= calculate_running_total(5, 4); ?></td></tr>
</table>
```

ERGEBNIS

ITEM	PRICE	QTY	RUNNING TOTAL
Mints	\$2	5	\$12
Toffee	\$3	5	\$15
Fudge	\$5	4	\$19

1. Eine Variable `$tax_rate` wird im globalen Gültigkeitsbereich erstellt.

2. `calculate_running_total()` erstellt eine laufende Summe. Abbildung 10.10

3. Mit dem Schlüsselwort `global` kann die Funktion auf den Wert in der globalen Variablen `$tax_rate` zugreifen und diesen aktualisieren.

4. Das Schlüsselwort `static` besagt, dass die Variable `$running_total` (und ihr Wert) nicht gelöscht werden darf, wenn die Funktion beendet ist. (Bei ihrer Erstellung erhält sie den Anfangswert 0.)

5. `$total` enthält den Preis eines Produkts multipliziert mit der vom Kunden gewünschten Menge.

6. `$tax` enthält den Steuerbetrag, der für diese Artikel fällig ist, wobei der Wert in der in Schritt 1 erstellten globalen Variablen `$tax_rate` verwendet wird.

7. `$running_total` enthält den:

- Wert in `$running_total`,
- plus den Wert in `$total`,
- plus den Wert in `$tax`.

8. Der Wert in `$running_total` wird zurückgegeben, aber nicht gelöscht, da es sich um eine statische Variable handelt.

9. Die Funktion wird dreimal aufgerufen. Jedes Mal wird die Summe für diesen Posten zur vorherigen Summe addiert.

FUNKTIONEN UND ZUSAMMENGESETZTE DATENTYPEN

Zusammengesetzte Datentypen (wie Arrays) können mehrere Werte speichern. Funktionen können einen zusammengesetzten Datentyp als Argument entgegennehmen und einen zusammengesetzten Datentyp zurückgeben.

ZUSAMMENGESETZTEN DATENTYP ALS ARGUMENT NUTZEN

Bei der Definition einer Funktion können die Parameter so notiert werden, dass sie entweder einen skalaren oder einen zusammengesetzten Datentyp annehmen können:

- **Skalare Datentypen** enthalten einen einzelnen Datenwert: eine Zeichenkette, eine Zahl oder einen booleschen Wert.
- **Zusammengesetzte Datentypen** können mehrere Datenelemente enthalten. Sie haben in Kapitel 1 Arrays kennengelernt und werden in Kapitel 4 einen weiteren zusammengesetzten Datentyp, die Objekte, kennenlernen.

Auf der rechten Seite sehen Sie ein Beispiel, in dem ein Array mit drei verschiedenen Wechselkursen als einzelner Parameter an die Funktion übergeben wird.

EINEN DATENTYP ALS RÜCKGABEWERT NUTZEN

Eine Funktion sollte nur eine einzige Aufgabe (nicht mehrere Aufgaben) ausführen, aber eine einzelne Aufgabe kann mehr als einen Wert erzeugen, der zurückgegeben werden muss.

Wenn Sie mehr als einen Wert aus einer Funktion zurückgeben wollen, müssen Sie ein Array oder ein Objekt in der Funktion erstellen und dieses dann zurückgeben. Der Grund ist, dass eine Funktion nur einen skalaren oder zusammengesetzten Wert zurückgeben kann.

Sobald der PHP-Interpreter eine Anweisung ausgeführt hat, die mit dem Schlüsselwort `return` beginnt, beendet er die Ausführung des Codes in der Funktion und kehrt zu der Codezeile zurück, die die Funktion aufgerufen hat (auch wenn es weitere Anweisungen in der Funktionsdefinition gibt, die noch nicht ausgeführt wurden).

Auf der rechten Seite berechnet die Funktion `calculate_prices()` drei Preise für einen Artikel in drei Währungen und gibt die Preise als Array zurück.

MEHRERE WERTE AKZEPTIEREN UND ZURÜCKGEBEN

PHP

section_a/c03/functions-with-multiple-values.php

```
<?php  
① $us_price = 4;  
    $rates = [  
        'uk' => 0.81,  
        'eu' => 0.93,  
        'jp' => 113.21,  
    ];  
  
③ function calculate_prices($usd, $exchange_rates)  
  
    $prices = [  
        'pound' => $usd * $exchange_rates['uk'],  
        'euro'  => $usd * $exchange_rates['eu'],  
        'yen'   => $usd * $exchange_rates['jp'],  
    ];  
    ⑤ return $prices;  
}  
  
⑥ $global_prices = calculate_prices($us_price, $rates);  
?> ...  
<h2>Chocolates</h2>  
⑦ <p>US $<?= $us_price ?></p>  
⑧ <p>(UK &pound; <?= $global_prices['pound'] ?> |  
    EU &euro; <?= $global_prices['euro'] ?> |  
    JP &yen; <?= $global_prices['yen'] ?>)</p>
```

ERGEBNIS



1. Die Variable `$us_price` enthält den Dollarpreis eines Artikels.
 2. Die Variable `$rates` speichert ein assoziatives Array von drei Wechselkursen.
 3. Die Funktion `calculate_prices()` berechnet die Preise eines Artikels in drei Währungen und gibt diese Preise dann als Array zurück. Sie hat zwei Parameter: den Preis in US-Dollar und das Array mit den Wechselkursen.
 4. Es wird ein Array erstellt und in der Variablen `$prices` gespeichert. Das erste Element enthält den Preis für Großbritannien, der durch Multiplikation des US-Preises mit dem Wechselkurs für Großbritannien berechnet wird. Dann werden die Preise für die EU und Japan zu dem Array hinzugefügt.
 5. Die Funktion gibt das Array mit den drei neuen Preisen zurück.
 6. Die Funktion wird aufgerufen, und das Array, das sie zurückgibt, wird in der Variablen `$global_prices` gespeichert.
 7. Der US-Preis wird über die Variable aus Schritt 1 ausgegeben.
 8. Andere Preise aus dem in Schritt 6 erstellten Array werden angezeigt.
- Probieren Sie es: Fügen Sie einen Wechselkurs von 1,32 für australische Dollar hinzu.

DEKLARATIONEN VON ARGUMENTEN UND RÜCKGABETYPEN

Bei der Definition einer Funktion können Sie angeben, welchen Datentyp die einzelnen Argumente haben sollen und welchen Datentyp die Funktion zurückgeben soll.

Einige Aufgaben erfordern Daten von einem bestimmten Typ.

So benötigen beispielsweise Funktionen, die arithmetische Berechnungen durchführen, Zahlen als Argumente, und Funktionen, die Text verarbeiten, benötigen Zeichenketten.

Eine Funktionsdefinition kann die Datentypen angeben, die die einzelnen Parameter erwarten, und den Datentyp, den die Funktion zurückgeben soll. Dies hilft bei der Programmierung, da die erste Zeile der Funktionsdefinition deutlich zeigt, welchen Datentyp die einzelnen Argumente haben sollten und welchen Datentyp die Funktion zurückgeben sollte.

In der ersten Zeile der unten gezeigten Funktionsdefinition

- wird innerhalb der Klammern vor jedem Parameternamen ein **Argumenttyp** angegeben, um den Datentyp anzugeben, den das Argument verwenden soll.
- Nach den Klammern folgt ein Doppelpunkt, gefolgt von einem **Rückgabetyp**, um den Datentyp anzugeben, den die Funktion zurückgibt.

In diesem Fall sollten beide Argumente Ganzahlen sein, und der zurückgegebene Wert sollte ebenfalls eine Ganzzahl sein.

```
ARGUMENTTYP      ARGUMENTTYP      RÜCKGABETYP
          ↗          ↗          ↗
function calculate_total(int $price, int $quantity): int
{                                     ↗
    return $price * $quantity;       ↗
}                                     |
```

PARAMETER DOPPELPUNKT

Die Tabelle auf der rechten Seite zeigt die Datentypen, die in Argument- und Rückgabetypen-Deklarationen verwendet werden. In PHP 8 wurden hinzugefügt:

- **Union Types:** virtueller Typ, der die *Union* von zwei Typen darstellt. Jeder Typ wird durch ein Pipe-Symbol | getrennt. Zum Beispiel gibt `int | float` einen Integer oder einen Float an.
- **Mixed Types:** gibt an, dass ein Argument oder ein Rückgabetyp ein beliebiger Datentyp sein kann (man spricht von **Pseudo-Typ**, weil dieser Typ für Variablen nicht möglich ist).

DATENTYP	BESCHREIBUNG
string	String
int	Integer (Ganzzahl)
float	Fließkommazahl (dezimal)
bool	Boolescher Wert (wahr oder falsch / 0 oder 1)
array	Array
className	Objektklasse (siehe Kapitel 4)
mixed	Eine Mischung der oben genannten Datentypen (PHP 8)

TYPENDEKLARATIONEN VERWENDEN

PHP

section_a/c03/type-declarations.php

```
<?php  
$price    = 4;  
$quantity = 3;  
  
① function calculate_total(int $price, int $quantity): int  
{  
    return $price * $quantity;  
}  
  
$total = calculate_total($price, $quantity);  
?>  
<h1>The Candy Store</h1>  
<h2>Chocolates</h2>  
<p>Total $<?= $total ?></p>
```

ERGEBNIS



HINWEIS: Wenn ein Argument oder ein Rückgabetyp `null` sein kann (und kein Wert), können Sie ein Fragezeichen vor den Datentyp setzen.

Zum Beispiel gibt `?int` an, dass der Wert eine ganze Zahl oder `null` sein wird.

In PHP 8 könnte auch ein Union-Typ `int|null` verwendet werden.

In diesem Beispiel definiert die ① erste Zeile der Funktionsdefinition:

- Argumenttyp-Deklarationen für die Parameter `$price` und `$quantity`, um zu bestimmen, dass ihre Werte ganze Zahlen sein sollen.
- Eine Deklaration des Rückgabetyps, um zu bestimmen, dass die Funktion eine ganze Zahl zurückgeben soll.

Die Typdeklarationen haben keinen Einfluss auf die Funktionsweise dieses Beispiels; sie geben nur an, welche Datentypen die Argumente und Rückgabetypen haben sollten. Auf der nächsten Seite sehen Sie, wie man erzwingen kann, dass diese Werte die richtigen Datentypen verwenden.

Probieren Sie es: Ändern Sie den in den Variablen `$price` gespeicherten Wert so, dass er eine Zeichenkette statt einer ganzen Zahl ist, z.B.:

`$price = '1';`

Wenn Sie die Seite aktualisieren, sollten Sie dasselbe Ergebnis erhalten, da Sie strikte Typisierung verwenden müssen (siehe nächste Seite).

Probieren Sie es: Wenn Sie PHP 8 einsetzen, verwenden Sie Union-Typen, um anzugeben, dass die Werte Ganzzahlen oder Gleitkommazahlen sein können..

STRIKTE TYPISIERUNG AKTIVIEREN

Wenn eine Funktionsdefinition Argumente und/oder Rückgabewerte deklariert, können Sie den PHP-Interpreter anweisen, einen Fehler zu erzeugen, wenn eine Funktion mit dem falschen Datentyp aufgerufen wird oder den falschen Datentyp zurückgibt.

Wenn die Argumente einer Funktion vom falschen skalaren Datentyp sind, kann der PHP-Interpreter versuchen, sie in den erwarteten Datentyp zu konvertieren.

Um die Verarbeitung von Daten zu erleichtern, kann der PHP-Interpreter zum Beispiel konvertieren:

- die Zeichenkette '1' in die Ganzzahl 1
- den booleschen Wert `true` in die Ganzzahl 1
- den booleschen Wert `false` in die Ganzzahl 0
- die Ganzzahl 1 in den booleschen Wert `true`
- die Ganzzahl 0 in den booleschen Wert `false`

Dies sind Beispiele für **Type-Juggling**, das auf S. 60–61 vorgestellt wurde.

Sie können den PHP-Interpreter anweisen, strikte Typisierung zu aktivieren, sodass er einen Fehler auslöst, wenn eine Funktion

- mit einem Argument aufgerufen wird, das den falschen Datentyp hat (wenn bei der Deklaration ein Argumenttyp angegeben wurde)
- einen Wert mit dem falschen Datentyp zurückgibt (wenn bei der Deklaration ein Rückgabewert angegeben wurde)

Die ausgelösten Fehler können helfen, die Problemquelle im PHP-Code aufzuspüren, aber der PHP-Interpreter muss angewiesen werden, die Typen in der Seite, die die Funktion **aufruft**, mithilfe des unten stehenden `declare`-Konstrukts zu überprüfen.

Dies **muss** die erste Anweisung in der Seite sein, und sie aktiviert die strikte Typisierung nur für Funktionen, die in dieser Seite aufgerufen werden.

```
STRIKTE TYPEN      EIN  
declare(strict_types = 1);
```

Kapitel 10 beschäftigt sich mit der Fehlerbehandlung und -behebung, aber vielleicht haben Sie bemerkt, dass es im Download-Code mehrere Dateien `.htaccess` gibt. Diese Dateien steuern die Einstellungen des Webservers, z.B. ob Fehler in der HTML-Seite, die der Webserver an den Browser zurückschickt, gemeldet werden sollen oder nicht. Wenn Sie diese Dateien nicht sehen können, liegt es daran, dass sie vom Betriebssystem als versteckte Dateien behandelt werden (siehe S. 196).

Manchmal werden Funktionsdefinitionen in einer `Include`-Datei abgelegt, damit sie von mehreren Seiten einer Site aufgerufen werden können (wie Sie in Kapitel 6 sehen werden). Die strikte Typisierung muss in einer Datei, die nur Funktionsdefinitionen enthält, nicht aktiviert werden, aber sie muss auf den Seiten, die die Funktionen aufrufen, aktiviert werden, wenn Sie möchten, dass der PHP-Interpreter einen Fehler ausgibt, sobald der falsche Datentyp verwendet wird.

STRIKTE TYPISIERUNG VERWENDEN

PHP

section_a/c03/strict-types.php

```
<?php  
① declare(strict_types = 1);  
  
② $price    = 4;  
    $quantity = 3;  
  
③ function calculate_total(int $price, int $quantity): int  
{  
    ④     return $price * $quantity;  
}  
  
⑤ $total = calculate_total($price, $quantity);  
?  
<h1>The Candy Store</h1>  
<h2>Chocolates</h2>  
⑥ <p>Total $<?= $total ?></p>
```

ERGEBNIS



Probieren Sie es: In Schritt 2 setzen Sie den Wert für die Variable `$price` auf eine Zeichenkette: `$price = '4';` Nach der Seitenaktualisierung erhalten Sie eine Fehlermeldung.

Probieren Sie es: In PHP 8 verwenden Sie in Schritt 2 für den Preis `4 . 5`, und in Schritt 3 geben Sie mit Union-Types an, dass Argumente und Rückgabewerte `int` oder `float` sein können.

Dieses Beispiel ist fast identisch mit dem auf der vorherigen Seite, aber die erste Anweisung verwendet strikte Typisierung, um einen Fehler anzuzeigen, wenn die Argumente oder der Rückgabewert den falschen Datentyp haben.

1. Das `declare`-Konstrukt aktiviert die strikte Typisierung für die Seite.
2. Zwei Variablen werden deklariert, um einen Preis und eine Menge zu speichern.
3. Die Funktion `calculate_total()` wird definiert.
 - Die Deklarationen der Argumenttypen geben an, dass beide Parameter ganze Zahlen erwarten.
4. Die Deklaration des Rückgabetyps legt fest, dass die Funktion eine ganze Zahl zurückgibt.
5. Die Funktion multipliziert den Preis mit der Menge und gibt diesen Wert zurück.
6. Die Summe wird ausgeschrieben.

MEHRFACHE RETURN-ANWEISUNGEN

Funktionen können je nach dem Ergebnis einer bedingten Anweisung innerhalb der Funktion unterschiedliche Werte zurückgeben.

Mit bedingten Anweisungen können Sie festlegen, welcher Wert von einer Funktion zurückgegeben werden soll.

Die folgende Funktion gibt je nach dem als Argument übergebenen Wert unterschiedliche Meldungen zurück.

Sobald eine Funktion eine return-Anweisung verarbeitet hat, kehrt der PHP-Interpreter zu der Codezeile zurück, die die Funktion aufgerufen hat. Keine der nachfolgenden Anweisungen in dieser Funktion wird ausgeführt.

Die unten stehende Funktion enthält drei return-Anweisungen.

1. Eine Bedingung prüft, ob der Wert im Parameter \$stock 10 oder mehr beträgt. Ist dies der Fall, wird die erste return-Anweisung verarbeitet, und es werden keine weiteren Anweisungen in der Funktion ausgeführt.
2. Ist der Wert größer als 0 und kleiner als 10, wird die zweite return-Anweisung verarbeitet, und kein weiterer Code in der Funktion wird ausgeführt.
3. Während die Funktion ausgeführt wird, muss \$stock einen Wert von 0 haben, damit die letzte return-Anweisung verarbeitet wird.

```
function get_stock_message($stock)
{
    if ($stock >= 10) {
        return 'Good availability';
    }
    if ($stock > 0 && $stock < 10) {
        return 'Low stock';
    }
    return 'Out of stock';
}
```

MEHRERE RÜCKGABEN IN EINER FUNKTION VERWENDEN

PHP

section_a/c03/multiple-return-statements.php

```
<?php
① $stock = 25;

② function get_stock_message($stock)
{
    if ($stock >= 10) {
        return 'Good availability';
    }
    if ($stock > 0 && $stock < 10) {
        return 'Low stock';
    }
    ⑤ return 'Out of stock';
}
?>
<h1>The Candy Store</h1>
<h2>Chocolates</h2>
⑥ <p><?= get_stock_message($stock) ?></p>
```

ERGEBNIS



1. Die Variable \$stock enthält den Lagerbestand.

2. Die Funktion get_stock_message() überprüft den Lagerbestand und gibt eine von drei Meldungen zurück.

3. Eine bedingte Anweisung prüft, ob der Bestand größer oder gleich 10 ist. Ist dies der Fall, wird die erste return-Anweisung verarbeitet. Sie gibt die Meldung Good availability zurück, und der weitere Code der Funktion wird nicht ausgeführt.

4. Ist der Bestand nicht größer als 10, wird die Funktion weiter ausgeführt, und die nächste Bedingung prüft, ob der Bestand größer als 0 und kleiner als 10 ist. Ist dies der Fall, wird die zweite return-Anweisung verarbeitet. Sie gibt die Meldung Low stock zurück, und der Code der Funktion wird nicht weiter ausgeführt.

5. Wenn die Funktion weiterhin ausgeführt wird, kann es keinen Lagerbestand geben, sodass die letzte return-Anweisung die Meldung Out of stock zurückgibt.

6. Die Funktion wird aufgerufen, und der zurückgegebene Wert wird auf der Seite ausgegeben.

Probieren Sie es: Ändern Sie in Schritt 1 den Lagerbestand auf 8. Sie sollten die Meldung Low stock sehen.

OPTIONALE PARAMETER UND STANDARDWERTE

Der Parameter einer Funktion kann optional sein. Dazu geben Sie ihm einen Standardwert, der verwendet wird, wenn kein Wert angegeben wird. Optionale Parameter erscheinen in der Regel nach erforderlichen Parametern.

Für manche Aufgaben können optionale Informationen nötig sein. Diese Daten sind nicht *erforderlich*, damit die Funktion ihre Aufgabe erfüllen kann, aber ein Wert kann angegeben werden, wenn die Funktion aufgerufen wird.

Um einen Parameter optional zu machen, geben Sie ihm einen **Standardwert**, der verwendet wird, wenn die Funktion aufgerufen wird, ohne einen Wert für diesen Parameter anzugeben.

Der Standardwert wird in der Funktionsdefinition nach dem Parameternamen angegeben. Die Syntax ist dieselbe wie beim Zuweisen eines Werts zu einer Variablen.

```
function calculate_cost($cost, $quantity, $discount = 0)
{
    $cost = $cost * $quantity;
    return $cost - $discount;
}

$cost = calculate_cost(5, 3);
```

Wenn Sie die Funktionsweise einer Funktion dokumentieren, werden alle optionalen Parameter in eckige Klammern gesetzt. Wenn Sie die Funktion aufrufen, werden die eckigen Klammern nicht verwendet; sie zeigen nur an, dass die Funktion optional ist.

```
calculate_cost($cost, $quantity[, $discount])
```

Die folgende Funktion wird mit zwei Argumenten aufgerufen, sodass der letzte Parameter den Standardwert 0 erhält.

Optionale Parameter stehen nach den erforderlichen Parametern, denn bis PHP 8 mussten die Argumente beim Aufruf einer Funktion in der Reihenfolge angegeben werden, in der die Parameter in der Funktionsdefinition aufgeführt waren.

Sie werden im nächsten Kapitel mehr über die benannten Parameter von PHP 8 erfahren, aber es ist wahrscheinlich, dass die optionalen Parameter weiterhin nach den erforderlichen Parametern platziert werden.

HINWEIS: Das Komma vor den optionalen Parametern steht in den eckigen Klammern, weil das Setzen eines Kommas nach dem letzten Argument beim Aufruf einer Funktion bis PHP 8 (das Kommas am Ende erlaubt) einen Fehler verursachte.

OPTIONALER PARAMETER IN ECKIGEN KLAMMERN

STANDARDWERTE FÜR PARAMETER VERWENDEN

PHP

section_a/c03/default-values-for-parameters.php

```
<?php  
① function calculate_cost($cost, $quantity, $discount = 0)  
{  
    $cost = $cost * $quantity;  
    return $cost - $discount;  
}  
?>  
<h1>The Candy Store</h1>  
<h2>Chocolates</h2>  
② <p>Dark chocolate $<?= calculate_cost(5, 10, 5) ?></p>  
③ <p>Milk chocolate $<?= calculate_cost(3, 4) ?></p>  
④ <p>White chocolate $<?= calculate_cost(4, 15, 20) ?></p>
```

ERGEBNIS



1. Die Funktion `calculate_cost()` berechnet die Preis für einen oder mehrere Artikel auf der Grundlage von drei Informationen:

- Preis
- Menge
- Rabatt

Beim Aufruf der Funktion ist das letzte Argument optional, da der Parameter mit dem Standardwert 0 belegt ist. Die Funktion wird dann dreimal aufgerufen.

2. Beim ersten Aufruf erhält sie den Preis 5, die Menge 10 und den Rabatt 5, sodass die Funktion 5 von der Gesamtsumme 50 abzieht und 45 zurückgibt.

3. Beim zweiten Aufruf der Funktion werden die Preis mit 3 und die Menge mit 4 angegeben, aber kein Rabatt, sodass der Standardwert 0 verwendet wird. Als Ergebnis liefert die Funktion Preis von 12.

4. Beim dritten Aufruf erhält die Funktion den Preis 4, die Menge 15 und den Rabatt 20. Wie bei Schritt 2 wird ein Rabatt (diesmal 20) von den Preis (60) abgezogen, um einen Wert von 40 zu erhalten.

Probieren Sie es: Ändern Sie in Schritt 1 den Standardrabatt auf 2 und in Schritt 2 den Rabatt auf 7.

BENANNTE ARGUMENTE

Wenn Sie eine Funktion in PHP 8 aufrufen, können Sie die Parameternamen vor die Argumente setzen. Das bedeutet, dass die Argumente nicht in der Reihenfolge angegeben werden müssen, in der die Parameternamen in der Funktionsdefinition erscheinen.

In PHP 8 können Sie beim Aufruf einer Funktion Parameternamen vor den Argumenten hinzufügen. Diese werden **benannte Argumente** (oder benannte Parameter) genannt. Sie

- geben eindeutig an, was jedes Argument tut,
- können optionale Argumente auslassen, ohne einen Standardwert anzugeben oder leere Anführungszeichen zu verwenden (siehe unten).

Die Funktionsdefinition ändert sich nicht, nur die Art und Weise, wie die Argumente beim Aufruf der Funktion bereitgestellt werden. Das Beispiel auf der rechten Seite hat vier Parameter:

- `$cost` (erforderlich) ist der Preis für einen Artikel
- `$quantity` (erforderlich) ist die Anzahl dieses Artikels
- `$discount` (optional) ist der zu rabattierende Betrag
- `$tax` (optional) ist der Prozentsatz der fälligen Steuer

Wenn eine Funktion ohne benannte Argumente aufgerufen wird, müssen die Argumente in der gleichen Reihenfolge wie die Parameter in der Funktionsdefinition stehen:

```
calculate_cost(5, 10, 0, 5); or calculate_cost(5, 10, '', 5);
```

Bei der Verwendung von benannten Argumenten wird der Name durch einen Doppelpunkt vom Argument getrennt, und die Argumente können in beliebiger Reihenfolge stehen.

```
calculate_cost(quantity: 10, cost: 5, tax: 5);
```

Argumente ohne Parameternamen können vor benannten Argumenten erscheinen, wenn sie in der gleichen Reihenfolge wie die Parameter in der Funktionsdefinition erscheinen.

```
calculate_cost(5, 10, tax: 5);
```

Wenn ein Argument seinen Standardwert verwenden soll (der in der Funktionsdefinition angegeben ist), müssen Sie keinen Wert angeben oder leere Anführungszeichen verwenden:

Nachfolgend werden die ersten beiden Werte für Preis und Menge verwendet; anschließend wird die Steuer angegeben. Beachten Sie, dass der Wert für den Rabatt nicht angegeben ist.

BENANNTE ARGUMENTE VERWENDEN

PHP

section_a/c03/named-arguments-in-php-8.php

```
<?php  
① function calculate_cost($cost, $quantity, $discount = 0, $tax = 20,)  
{  
    $cost = $cost * $quantity;  
    $tax = $cost * ($tax / 100);  
    return ($cost + $tax) - $discount;  
}  
?  
<h1>The Candy Store</h1>  
<h2>Chocolates</h2>  
② <p>Dark chocolate $<?= calculate_cost(quantity: 10, cost: 5, tax: 5, discount: 2); ?></p>  
③ <p>Milk chocolate $<?= calculate_cost(quantity: 10, cost: 5, tax: 5); ?></p>  
④ <p>White chocolate $<?= calculate_cost(5, 10, tax: 5); ?></p>
```

1. Die Funktion

`calculate_cost()` berechnet die Preis für einen oder mehrere Artikel auf der Grundlage von vier Informationen:

- Preis (erforderlich)
- Menge (erforderlich)
- Rabatt (optional – Standardwert 0)
- Steuer (optional – Standardwert 20%)

Da in diesem Beispiel PHP 8 verwendet wird, kann nach dem letzten Parameter in der Funktionsdefinition (und nicht nur zwischen den Parametern) ein nachgestelltes Komma eingefügt werden. Dadurch wird die Einheitlichkeit des Codes verbessert (da nach jedem Parameter ein Komma stehen kann).

Nachdem die Funktion definiert wurde, wird sie dreimal aufgerufen.

ERGEBNIS



2. Die Funktion wird mit vier benannten Argumenten aufgerufen. Da sie benannt sind, können die Argumente in beliebiger Reihenfolge auftreten.

3. Benannte Argumente werden für Preis, Menge und Steuer verwendet. Es wird der Standardrabatt verwendet.

4. Die ersten beiden Werte verwenden keine benannten Argumente, sodass sie für die ersten beiden Parameter (`$cost` und `$quantity`) verwendet werden. Für `$discount` wird kein Wert angegeben, sodass das letzte Argument benannt werden muss, damit es für den Parameter `$tax` verwendet werden kann.

SO ERSTELLEN SIE EINE FUNKTION

Diese vier Schritte helfen Ihnen beim Schreiben von Funktionen.

1. BESCHREIBEN SIE KURZ DIE AUFGABE

Verwenden Sie eine Kombination aus dem Zweck der Funktion (z.B. abrufen, berechnen, aktualisieren oder speichern), gefolgt von der Art der Daten, mit denen sie arbeitet. Daraus erstellen Sie den Name der Funktion.

Der Funktionsname ändert sich nie.

2. WELCHE DATEN WERDEN FÜR DIE AUFGABE BENÖTIGT?

Jedes Datenelement wird zu einem Parameter.

Die an den Parameter (die Argumente) übergebenen Werte können sich bei jedem Funktionsaufruf ändern.

3. WELCHE ANWEISUNGEN MUSS DIE FUNKTION BEFOLGEN, UM DIE AUFGABE ZU ERFÜLLEN?

Die Anweisungen werden durch Befehle innerhalb der geschweiften Klammern dargestellt.

Die Anweisungen sind bei jedem Funktionsaufruf dieselben.

4. WELCHES ERGEBNIS ERWARTEN SIE?

Dies ist der Wert, der von der Funktion zurückgegeben wird. Es gilt als gute Praxis, dass eine Funktion einen Wert zurückgibt. Wenn Sie eine Aufgabe ausführen, bei der kein neuer Wert berechnet oder keine Information abgerufen wird, gibt die Funktion oft true oder false zurück, um anzudeuten, ob sie funktioniert hat oder nicht.

Der zurückgegebene Wert kann sich jedes Mal ändern, wenn der Funktion neue Werte zur Verfügung gestellt werden.

FUNCTIONS

TOTAL VALUE OF STOCK

get_total_value()

NEED price
quantity

STEPS price × quantity

RETURN value

TOTAL TAX DUE

get_tax_due()

NEED price
quantity
tax rate

STEPS price × quantity

divide by 100

multiply by tax rate

RETURN total value



WARUM FUNKTIONEN VERWENDEN?

Es bietet sich an, den Code für eine Aufgabe in einer Funktion festzuhalten. Diese Seite listet die Gründe auf.



WIEDERVERWENDBARKEIT

Wie Sie in diesem Kapitel bereits mehrfach gesehen haben, müssen Sie den Code für eine Seite, die dieselbe Aufgabe mehrfach ausführen muss (z.B. die Berechnung von Artikelpreisen), nur einmal schreiben. Sobald die Seite die Aufgabe erfüllen muss, ruft sie die Funktion auf und über gibt ihr die dazu benötigten Werte.

WARTBARKEIT

Wenn Sie feststellen, dass sich die für die Ausführung einer Aufgabe erforderlichen Anweisungen ändern, müssen Sie nur den Code in der Funktionsdefinition ändern (Sie müssen nicht jedes Mal Änderungen vornehmen, wenn die Aufgabe ausgeführt wird). Sobald die Funktionsdefinition aktualisiert wurde, wird bei jedem Aufruf der Funktion der aktualisierte Code verwendet.

ORGANISATION

Indem Sie den Code für jede Aufgabe in einer Funktion unterbringen, wird es einfacher, alle zur Ausführung der Aufgabe nötigen Anweisungen zu finden.

TESTBARKEIT

Indem Sie Ihren Code in die einzelnen Aufgaben aufteilen, können Sie jede einzelne Aufgabe für sich testen und dadurch Probleme leichter eingrenzen.

FUNKTIONEN DOKUMENTIEREN

Oft müssen Sie Funktionen verwenden, die Sie nicht selbst geschrieben haben, z.B. bei der Arbeit an einer großen Website in einem Entwicklungsteam. Die Dokumentation hilft den Beteiligten zu verstehen, wie sie diese Funktionen verwenden können.

Um eine Funktion in Ihrer PHP-Seite zu verwenden, müssen Sie nicht verstehen, wie die Anweisungen in den geschweiften Klammern Ihre Aufgabe erfüllen, Sie müssen nur Folgendes wissen:

- was die Funktion tun soll
- den Name der Funktion
- die Parameter, die sie benötigt
- was sie zurückgeben soll

Auf der rechten Seite sehen Sie eine Seite der Website [PHP . net](http://PHP.net). Das ist die offizielle Homepage von PHP mit der Dokumentation für diese Sprache – eine sehr nützliche Ressource, wenn Sie PHP erlernen wollen.

Rechts sehen Sie eine Funktion, die die Anzahl der Zeichen in einer Zeichenkette bestimmt. Diese Seite ist ein typisches Beispiel dafür, wie Funktionen dokumentiert werden. Normalerweise sehen Sie:

- Name und Beschreibung der Funktion
- die Syntax für den Aufruf der Funktion und die Verwendung ihrer Parameter (Argumente und Rückgabetypen können angezeigt werden)
- eine Beschreibung der Parameter
- welche(n) Wert(e) die Funktion zurückgibt
- ein Beispiel, wie Sie die Funktion verwenden können

Es ist wichtig, zwischen zwei Arten von Funktionen zu unterscheiden:

- **Benutzerdefinierte Funktionen** werden in einer PHP-Datei von einer Programmiererin definiert, die PHP-Sprache verwendet. (Alle Funktionen in diesem Kapitel sind benutzerdefinierte Funktionen.)
- **Eingebaute Funktionen** werden von den Entwicklern der PHP-Sprache definiert, und ihre Definitionen werden im PHP-Interpreter implementiert. Das bedeutet, dass jeder diese Funktionen aufrufen kann, ohne die Funktionsdefinition in eine Seite einzufügen zu müssen.

Eingebaute Funktionen führen Aufgaben aus, die Programmiererinnen und Programmierer beim Schreiben von PHP-Code häufig benötigen. Sie ersparen es ihnen, für diese Aufgaben ihren eigenen Code zu schreiben und das Rad jedes Mal neu zu erfinden. Mehr über die eingebauten Funktionen erfahren Sie in Kapitel 5.

1 strlen

(PHP 4, PHP 5, PHP 7, PHP 8)
strlen -- Get string length

2 Description

`strlen(string $string): int`

Returns the length of the given **string**.

3 Parameters

string

The string being measured for length.

4 Return Values

The length of the **string** on success, and 0 if the **string** is empty.

5 Examples

Example #1 A `strlen()` example

```
<?php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
?>
```

Notes

Note:

`strlen()` returns the number of bytes rather than the number of characters in a string.

Note:

`strlen()` returns `null` when executed on arrays, and an `E_WARNING` level error is emitted.

Change language: English
[Submit a Pull Request](#) [Report a Bug](#)

String Functions

addcslashes
addslashes
bin2hex
chr
clen
convert_crypt_link
convert_hexcolor
convert_uuencode
count_chars
crc32
crypt
echo
explode
fopen
get_html_translation_table
hebrev
hex2bin
html_entity_decode
htmlspecialchars
htmlspecialchars_decode
htmlspecialchars_entิต
htmlspecialchars_entත
htmlspecialchars_entລ
htmlspecialchars_entວ
htmlspecialchars_entໝ
join
ltrim
levenshtein
localeconv
ltrim
md5_file
nl2br
number_format
ord
parse_str
prev
print_r
quoted_printable_decode
quoted_printable_encode
quotemeta
rtrim
setlocale
sha1_file
strchr
similar_text
soundex
strcut
sscanf
str_contains
str_endwith
str_getcsv
str_replace
str_pad
str_repeat
str_replace



BEISPIEL



Dieses Beispiel zeigt eine Seite zur Überwachung der Lagerbestände in einem Süßwarenladen.

Es wird ein assoziatives Array erstellt, das die Namen der Produkte, die der Laden verkauft, und die Lagerbestände für jedes Produkt enthält. Diese Werte werden in den ersten beiden Spalten der Tabelle angezeigt.

Dann werden drei Funktionen für die Werte in den nächsten drei Spalten erstellt:

- Die erste Funktion prüft die Lagerbestände und erstellt eine Meldung, die angibt, ob weitere Bestände bestellt werden müssen.
- Die zweite Funktion errechnet den Gesamtwert des Lagerbestands für jeden verkauften Artikel.
- Die dritte Funktion berechnet, wie viel Steuer fällig wird, wenn der gesamte Restbestand verkauft wurde.

```

<?php
① declare(strict_types = 1);
    $candy = [
        'Toffee' => ['price' => 3.00, 'stock' => 12],
        'Mints'  => ['price' => 2.00, 'stock' => 26],
        'Fudge'   => ['price' => 4.00, 'stock' => 8],
    ];
③ $tax = 20;

④ function get_reorder_message(int $stock): string
{
⑤     return ($stock < 10) ? 'Yes' : 'No';
}

⑥ function get_total_value(float $price, int $quantity): float
{
⑦     return $price * $quantity;
}

⑧ function get_tax_due(float $price, int $quantity, int $tax = 0): float
{
⑨     return ($price * $quantity) * ($tax / 100);
}
?>

```

1. Strikte Typisierung ist aktiviert.
2. Ein mehrdimensionales Array wird erstellt (siehe S.44–45) und in der Variablen \$candy gespeichert.
 - Die Schlüssel sind die Namen der verkauften Süßigkeiten.
 - Die Werte sind Arrays, die Preis und verfügbaren Lagerbestand des jeweiligen Produkts enthalten.
3. Eine globale Variable für den Steuersatz wird deklariert.
4. Die Funktion get_reorder_message() wird definiert. Sie hat einen Parameter, den aktuellen Lagerbestand für ein Produkt (int). Sie gibt eine Nachricht (Zeichenkette) zurück, die besagt, ob der Artikel nachbestellt werden soll oder nicht.
5. Mit einem ternären Operator wird eine Nachricht zurückgegeben. Die Bedingung prüft, ob der Lagerbestand weniger als 10 beträgt:
 - Ist dies der Fall, gibt die Funktion Yes zurück.
 - Andernfalls gibt sie No zurück.

6. Die Funktion get_total_value() mit zwei Parametern wird definiert:
 - Preis des Produkts (float)
 - verfügbare Menge dieses Produkts (int)

Sie gibt einen float mit dem Gesamtwert des Bestands für dieses Produkt zurück (hier ist auch ein int eine gültige Zahl).
7. Die Funktion gibt den Preis des Produkts multipliziert mit der Menge des verfügbaren Bestands zurück.
8. Die Funktion get_tax_due() wird definiert. Sie hat drei Parameter:
 - den Preis des Produkts (float)
 - die verfügbare Menge dieses Produkts (int)
 - den Steuersatz in Prozent mit einem Standardwert von 0% (int)

Der Rückgabewert ist ein Fließkommawert für den Gesamtbetrag der Steuer, die fällig wird, wenn diese Produkte verkauft werden.

```

<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Stock Control</h2>
        <table>
            <tr>
                <th>Candy</th><th>Stock</th><th>Re-order</th><th>Total value</th><th>Tax due</th>
            </tr>
            <?php foreach ($candy as $product_name => $data) { ?>
            <tr>
                <td><?= $product_name ?></td>
                <td><?= $data['stock'] ?></td>
                <td><?= get_reorder_message($data['stock']) ?></td>
                <td>$<?= get_total_value($data['price'], $data['stock']) ?></td>
                <td>$<?= get_tax_due($data['price'], $data['stock'], $tax) ?></td>
            </tr>
            <?php } ?>
        </table>
    </body>
</html>

```

- 9.** Der Gesamtbetrag der fälligen Steuer wird zurückgegeben. Zur Berechnung wird der Gesamtwert des Bestands (Preis eines Artikels multipliziert mit der verfügbaren Menge) mit dem Steuerprozentsatz (Steuersatz geteilt durch 100) multipliziert.
- 10.** Eine foreach-Schleife arbeitet die Produkte in dem in \$candy gespeicherten Array durch. In den Klammern finden Sie:
- \$candy ist die Variable, die das Array aus Schritt 2 speichert.
 - \$product_name ist der Name der Variablen, die den Schlüssel für das aktuelle Element des Arrays enthält (der Name des Produkts: toffee, mints oder fudge).
 - \$data ist die Variable, die den Wert für das aktuelle Element darstellt. Dies ist das Array, das den Preis und den Lagerbestand des Produkts speichert.
- 11.** Eine Tabellenzeile wird erstellt, und der Name des Produkts, das die Schleife gerade verarbeitet, wird in ein <td>-Element geschrieben.
- 12.** \$data enthält ein Array mit dem Preis und dem Lagerbestand dieses Produkts; der Lagerbestand wird in der nächsten Tabellenzelle ausgegeben.
- 13.** Die Funktion get_reorder_message() wird aufgerufen. Der Lagerbestand wird als Argument übergeben. Der zurückgegebene Wert wird in der Tabelle angezeigt.
- 14.** Die Funktion get_total_value() wird aufgerufen.
- Der erste Parameter ist der Preis des Produkts.
 - Der zweite Parameter ist die verfügbare Menge.
 - Der zurückgegebene Wert wird in die Tabelle geschrieben.
- 15.** Die Funktion get_tax_due() wird aufgerufen.
- Der erste Parameter ist der Preis des Produkts.
 - Der zweite Parameter ist die verfügbare Menge.
 - Der dritte Parameter ist der in Schritt 3 gespeicherte Steuersatz.
 - Der zurückgegebene Wert wird in die Tabelle geschrieben.
- 16.** Die schließende geschweifte Klammer beendet den Code-Block, und die Schleife wird für jedes Element des Arrays wiederholt.

ZUSAMMENFASSUNG

FUNKTIONEN

- **Funktionsdefinitionen** geben einer Funktion einen Namen und verwenden einen Code-Block, um die Anweisungen zur Ausführung einer Aufgabe zu speichern.
- Der Aufruf einer Funktion weist den PHP-Interpreter an, diese Anweisungen auszuführen, um die Aufgabe zu erfüllen.
- Das Schlüsselwort `return` sendet Daten von einer Funktion zurück.
- Parameter stehen für die Daten, die eine Funktion benötigt, um ihre Aufgabe zu erfüllen. Parameternamen verhalten sich in der Funktion wie Variablen.
- Wenn eine Funktion ausgeführt wurde, werden die Parameter und alle in der Funktion deklarierten Variablen gelöscht.
- Wenn eine Funktion aufgerufen wird, werden die für die Parameter verwendeten Werte als Argumente bezeichnet.
- Typdeklarationen geben den Datentyp für Argumente an.
- Rückgabetypen geben den Datentyp an, den eine Funktion zurückgibt.
- Wenn ein Parameter optional ist, erhält er einen Standardwert.

4

KLASSEN & OBJEKTE

Objekte fassen eine Reihe von Variablen und Funktionen zusammen, die Dinge des täglichen Lebens repräsentieren, z.B. Nachrichtenartikel, Waren oder Benutzer einer Website.

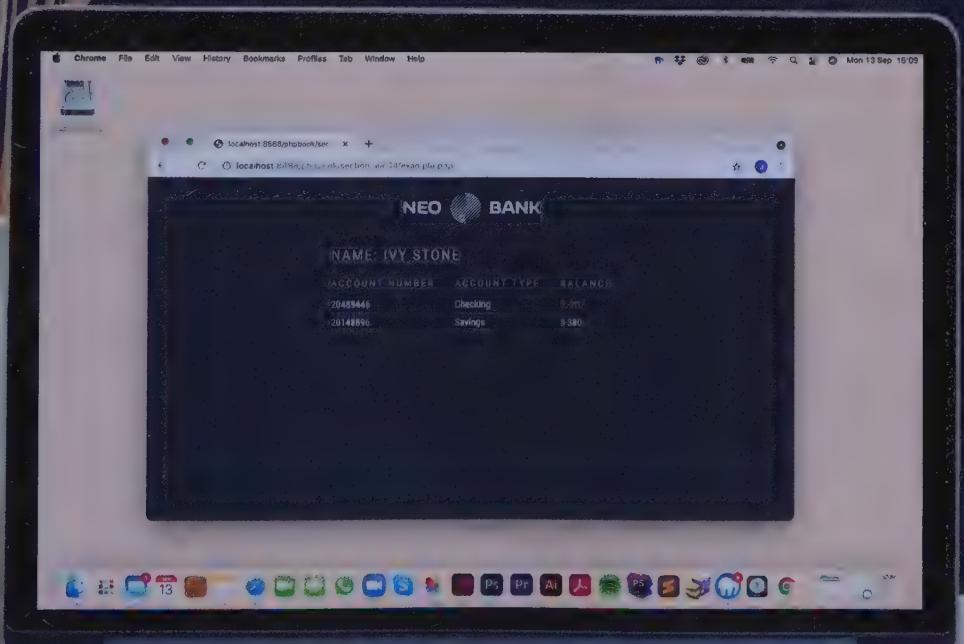
- In Kapitel 2 haben Sie gesehen, dass Variablen einzelne Informationen speichern können. Wenn eine Variable in einem Objekt verwendet wird, spricht man von einer **Eigenschaft** des Objekts.
- In Kapitel 3 haben Sie gesehen, dass Funktionen für eine Aufgabe stehen können, die Ihr Code ausführen soll. Wenn eine Funktion in einem Objekt verwendet wird, nennt man sie eine **Methode** des Objekts.

In Websites muss oft eine Vielzahl gleicher Dinge dargestellt werden. Eine Nachrichtensite veröffentlicht zahlreiche Nachrichtenartikel, ein Shop verkauft viele Produkte, und eine Site, auf der sich Benutzer registrieren können, hat viele Mitglieder. All dies kann im Code durch ein **Objekt** dargestellt werden.

PHP verwendet eine sogenannte **Klasse** als Vorlage für die Erstellung von Objekten, die eine bestimmte Art von Dingen repräsentieren. Zum Beispiel kann eine Klasse verwendet werden, um Objekte für Produkte zu erstellen, und eine andere, um Objekte für Mitglieder zu erstellen. Jedes Objekt, das mit einer Klasse erstellt wird, erhält automatisch die Eigenschaften und Methoden, die in dieser Klasse definiert sind.

Objekte und Klassen helfen, den Code zu organisieren, und machen ihn leichter verständlich. Es ist wichtig zu lernen, wie Objekte funktionieren, da der PHP-Interpreter über mehrere eingebaute Objekte verfügt, die Sie im nächsten Teil dieses Buchs kennenlernen werden.

Auf den ersten Seiten dieses Kapitels werden die Konzepte hinter Objekten und ihrer Verwendung vorgestellt. Anschließend lernen Sie den Code kennen, der zum Erstellen und Verwenden von Objekten und Klassen erforderlich ist.



WEBSITES ALS MODELLE

Modelle sind Darstellungen von den Dingen in der Welt um uns herum. Programmiererinnen und Programmierer erstellen Modelle unter Verwendung von Daten; dann führen sie mithilfe von Code Aufgaben aus, die die in diesen Modellen gespeicherten Daten manipulieren.

In Websites werden mit Daten Modelle für Dinge des täglichen Lebens erstellt. Man bezeichnet diese als verschiedene **Objektarten**, beispielsweise:

- Personen (z.B. Kunden oder Mitglieder einer Website)
- Produkte oder Dienstleistungen, die Besucher kaufen könnten (z.B. Bücher, Autos, Finanzprodukte oder TV-Abonnements)
- Dokumente, die traditionell gedruckt wurden (z.B. Nachrichtenartikel, Kalender oder Eintrittskarten)

Eine Bank benötigt zum Beispiel bestimmte Daten für ihre einzelnen Kunden:

- Vorname
- Nachname
- E-Mail
- Kennwort

Sie benötigt für jede Person die gleichen Daten, aber die Namen, E-Mail-Adressen und Passwörter sind für jeden Kunden unterschiedlich.

Sie benötigt auch die gleichen Daten für jedes Bankkonto, aber die Werte für die einzelnen Konten sind unterschiedlich, beispielsweise

- Kontonummer
- Art des Kontos
- Saldo

Die Bank kann mit diesen Daten Aufgaben durchführen. Zu den Aufgaben, die mit einem Bankkonto durchgeführt werden können, gehören beispielsweise

- den Kontostand prüfen
- eine Einzahlung vornehmen
- Geld abheben

Solche Aufgaben benötigen oder aktualisieren Daten, die in Variablen gespeichert sind. Wenn Sie z.B. Geld abheben oder eine Einzahlung vornehmen, wird der als Kontostand gespeicherte Betrag geändert.

Ähnlich könnten Aufgaben im Zusammenhang mit einem Kunden aussehen:

- Authentifizierung des Benutzers (Bestätigung, dass er derjenige ist, für den er sich ausgibt), indem überprüft wird, ob die angegebene E-Mail-Adresse und das Passwort mit den für ihn gespeicherten Daten übereinstimmen
- Ermittlung des vollständigen Namens (durch Kombination von Vor- und Nachname)

All dies wird in einem **Objekt** zusammengefasst:

- Variablen, die Daten speichern, die zur Erstellung des Modells eines Konzepts benötigt werden, z.B. eines Kunden oder eines Kontos
- Funktionen, die Aufgaben repräsentieren, die ein Objekttyp ausführen kann

Wenn das Foto auf der rechten Seite entfernt wird, könnten Sie anhand der Informationen in den Kästchen immer noch viel erkennen: die Objektarten, die Daten, die zur Darstellung der einzelnen Objekte erforderlich sind, und die Aufgaben, die die Objekte erfüllen können.

OBJEKTYP: KUNDE

DATEN	WERT
forename	Ivy
surname	Stone
email	ivy@eg.link
password	\$2y\$10\$MAdTTCAOMiOwhewg...

AUFGABE ZWECK

Namen ermitteln	vollständigen Namen abrufen
authentifizieren	Übereinstimmung von E-Mail und Passwort prüfen



OBJEKTYP: KONTO

DATEN	WERT
number	20489446
type	Checking
balance	1000.00

AUFGABE	VERWENDUNGSZWECK
Einzahlung	Geld einzahlen
Auszahlung	Geld abheben
Kontostand	Saldo abrufen

Die Informationsfelder zeigen zwei Objektarten: **Kunden** und **Konten**. Für jede Objektart muss die Website:

1. die zur Darstellung verwendeten Daten speichern

Die einzelnen gespeicherten Datenelemente sind für jeden Kunden oder jedes Konto gleich, aber die Werte, für diesen Kunden oder dieses Konto sind unterschiedlich.

2. dieselben Aufgaben mit diesem Objekttyp durchführen

Sie können für jeden Kunden die gleichen Aufgaben durchführen, und Sie können für jedes Konto die gleichen Aufgaben durchführen.

Diese Aufgaben können auf die Daten, die für jeden Kunden oder jedes Konto gespeichert sind, zugreifen oder sie ändern. Wenn beispielsweise eine Einzahlung auf ein Konto vorgenommen wird, wird der Wert, der den Kontostand repräsentiert, aktualisiert.

EIGENSCHAFTEN UND METHODEN

In einem Objekt werden die Variablen als **Eigenschaften** und die Funktionen als **Methoden** bezeichnet. Eigenschaften speichern die zur Erstellung des Modells eines Konzepts benötigten Daten. Methoden stellen die Aufgaben dar, die diese Objektart ausführen kann.

VARIABLEN:

DIE EIGENSCHAFTEN EINES OBJEKTS

In Kapitel 1 haben Sie gesehen, dass Variablen Daten speichern können, die sich bei jedem Seitenaufruf ändern. Werden Variablen innerhalb eines Objekts verwendet, werden sie als Objekteigenschaften bezeichnet.

Bei der Erstellung eines Objekts müssen Sie entscheiden, welche Daten Sie für diese Art Objekt benötigen, damit die Website ihre Aufgabe erfüllen kann.

Werden zum Beispiel Objekte zur Darstellung von Kunden verwendet, besitzt jedes Kundenobjekt

- die *gleichen* Eigenschaften wie Vorname, Nachname, E-Mail-Adresse und Passwort, aber
- *unterschiedliche* Werte für jeden Kunden.

Wenn Sie ein Objekt für ein Konto verwenden, würde jedes Kontoobjekt ebenfalls

- über die *gleichen* Eigenschaften für Kontonummer, Kontotyp und Saldo verfügen, aber
- über *unterschiedliche* Werte für jedes Konto.

Die Objekteigenschaften sind ein Satz Variablen, die die Merkmale beschreiben, die alle diese Objekte gemeinsam haben. Die für jede Eigenschaft gespeicherten Werte unterscheiden ein Objekt von einem anderen.

FUNKTIONEN:

DIE METHODEN EINES OBJEKTS

In Kapitel 3 haben Sie gesehen, dass PHP alle Anweisungen, die zur Ausführung einer Aufgabe erforderlich sind, in einer Funktion zusammenfassen kann. Werden Funktionen innerhalb eines Objekts verwendet, nennt man sie Methoden des Objekts.

Bei der Erstellung eines Objekts entscheidet der Programmierer, welche Aufgaben die Benutzer der Webseite mit den einzelnen Objektarten ausführen können, zum Beispiel:

- anhand der in den Objekteigenschaften gespeicherten Daten Auskunft über das Objekt erhalten
- die in einer oder mehreren Objekteigenschaften gespeicherten Werte ändern

Die Aufgaben, die Sie mit einem Konto ausführen können (Einzahlung vornehmen, Geld abheben oder Kontostand prüfen), gelten für jedes Konto: Alle Objekte für ein Konto verfügen über dieselben Methoden.

In ähnlicher Weise müssen Sie für jeden Kunden die gleichen Aufgaben durchführen (Authentifizierung, Abfrage des Namens), und somit verfügt jedes Objekt, das einen Kunden repräsentiert, über dieselben Methoden.

Auf der rechten Seite sehen Sie eine ähnliche Darstellung wie auf der vorherigen Seite, aber diesmal sind die Eigenschafts- und Methodennamen für zwei Kunden und zwei Kontoobjekte dargestellt.

OBJEKTYP: KUNDE

DATEN	WERT
forename	Ivy
surname	Stone
email	ivy@eg.link
password	\$2y\$10\$MAdTTCA0Mi0whewg...

AUFGABE ZWECK

getFullName()	Werte für die Eigenschaften forename und surname zurückgeben
authenticate()	email und password prüfen

OBJEKTYP: KUNDE

DATEN	WERT
forename	Emiko
surname	Ito
email	emi@eg.link
password	\$2y\$10\$NN5HEAD3atarECjRi...

AUFGABE ZWECK

getFullName()	Werte für die Eigenschaften forename und surname zurückgeben
authenticate()	email und password prüfen

OBJEKTYP: KONTO

DATEN	WERT
number	20489446
type	Checking
balance	1000.00

AUFGABE ZWECK

deposit()	Wert der Eigenschaft balance erhöhen
withdraw()	Wert der Eigenschaft balance verringern
getBalance()	Wert der Eigenschaft balance zurückgeben

OBJEKTYP: KONTO

DATEN	WERT
number	10937528
type	Savings
balance	2346.00

AUFGABE ZWECK

deposit()	Wert der Eigenschaft balance erhöhen
withdraw()	Wert der Eigenschaft balance verringern
getBalance()	Wert der Eigenschaft balance zurückgeben

OBJEKT-DATENTYP

Ein Objekt ist ein Beispiel für einen zusammengesetzten Datentyp, da es mehrere Werte speichern kann.

Wie Sie gesehen haben, gibt es in PHP verschiedene Datentypen:

- Skalare Datentypen enthalten einzelne Werte: Zeichenketten, Ganzzahlen, Gleitkommazahlen, boolesche Werte.
- Zusammengesetzte Datentypen enthalten mehrere Werte: Arrays und Objekte.

Unten sehen Sie zwei Darstellungen von Objekten für einen Kunden und sein Konto.

Eine Variable, die ein Objekt enthält, kann wie jede andere Variable benannt werden (in Kleinbuchstaben mit einem Unterstrich zur Trennung der einzelnen Wörter, wenn der Name mehrere Wörter enthält). Zum Beispiel:

- Die Variable `$customer` kann ein Objekt für einen Kunden speichern.
- Die Variable `$account` kann ein Objekt für ein Konto speichern.

VARIABLENNNAME	CUSTOMER-OBJEKT	VARIABLENNNAME	ACCOUNT-OBJEKT
EIGENSCHAFT	WERT	EIGENSCHAFT	WERT
<code>\$customer</code>	<code>firstname</code> => Ivy	<code>\$account</code>	= <code>number</code> => 39488446
	<code>surname</code> => Stone	<code>type</code>	-> Checking
	<code>email</code> => ivy@eg.link	<code>balance</code>	-> 1000.00
	<code>password</code> => \$2y\$10\$MAdT...		

Oft sagt man, dass ein Objekt in einer Variablen gespeichert wird, aber wie Sie auf S. 530–531 erfahren, speichert die Variable in Wirklichkeit einen **Verweis** darauf, wo das Objekt im Speicher des PHP-Interpreters erstellt wurde.

Wenn eine Seite abgearbeitet ist und der PHP-Interpreter eine HTML-Seite an den Browser zurückgeschickt hat, vergisst er das Objekt (er löscht es aus seinem Speicher, genau wie er die in Variablen gespeicherten Werte verwirft).

KLASSEN ALS VORLAGEN FÜR OBJEKTE

Um Objekte zu erstellen, verwenden Sie eine Vorlage, die als **Klasse** bezeichnet wird. Eine Klassendefinition legt die Eigenschaftsnamen und Methoden fest, über die ein Objekttyp verfügt.

Eine Klassendefinition legt fest:

- Eigenschaftsnamen, die die Daten beschreiben, die für eine Objektart gespeichert werden müssen
 - Methoden, die Aufgaben definieren, die mit dieser Art von Objekt durchgeführt werden können
- Jedes Mal, wenn Sie ein Objekt mithilfe einer Klasse erstellen,
- geben Sie die Werte für die Eigenschaften an (diese Werte unterscheiden ein Objekt von einem anderen),
 - erhält das Objekt automatisch alle Methoden, die in der Klasse definiert wurden.

Jedes einzelne Objekt, das unter Verwendung der Klasse erstellt wird, wird als **Instanz** dieser Klasse bezeichnet.

Wenn Sie zum Beispiel ein Objekt für ein Bankkonto erstellen (wie in diesem Kapitel), geben Sie die Werte für die folgenden Eigenschaften an:

- \$number
- \$type
- \$balance

Und es erhält automatisch diese Methoden:

- deposit()
- withdraw()
- getBalance()

Manche Programmiererinnen und Programmierer verwenden die Begriffe »Klasse« und »Objekt« synonym, aber streng genommen ist eine Klasse eine Vorlage, die zur Erstellung eines Objekts verwendet wird.

SO ERSTELLEN UND VERWENDEN SIE OBJEKTE

Diese Seite erläutert die Schritte, die Sie erlernen müssen, um Klassen und Objekte zu erstellen und zu verwenden.

KLASSE ALS VORLAGE FÜR EIN OBJEKT DEFINIEREN

Siehe Seite 154

Eine Klasse ist eine Vorlage, die zur Erstellung eines Objekttyps verwendet wird.

Die Klasse definiert:

- die Eigenschaften, die die Daten speichern, die zur Darstellung dieser Objektart verwendet werden
- die Methoden, die die Anweisungen für die Aufgaben enthalten, die diese Objektart ausführen kann

Für jede Objektart, mit der die Website umgehen muss, wird eine neue Klasse erstellt.

EIN OBJEKT ERSTELLEN UND IN EINER VARIABLEN SPEICHERN

Siehe Seite 155

Ein Objekt wird erstellt durch:

- Angabe des Namens der Klasse, die als Vorlage für das Objekt verwendet werden soll
- Angabe von Werten für seine Eigenschaften

Das Objekt erhält automatisch die in der Klasse definierten Methoden.

Wenn ein Objekt erstellt wird, wird es in der Regel in einer Variablen gespeichert, damit es vom restlichen Code der Seite verwendet werden kann.

EIGENSCHAFTSWERTE EINSTELLEN UND DARAUF ZUGREIFEN

Siehe Seiten 156 bis 157

Sobald ein Objekt erstellt wurde, ist es möglich:

- Werte für seine Eigenschaften zu setzen
- auf die in seinen Eigenschaften gespeicherten Werte zuzugreifen und sie im restlichen Code der Seite zu verwenden

Jede Instanz eines Objekts speichert andere Werte in ihren Eigenschaften, da sie eine andere Instanz dieser Objektart repräsentiert (z.B. einen anderen Kunden oder ein anderes Konto).

METHODEN EINES OBJEKTS DEFINIEREN UND AUFRUFEN

Siehe Seiten 158 bis 159

Eine Methode definiert die Anweisungen, die für eine Aufgabe erforderlich sind, die ein Objekttyp ausführen kann. Sie werden genau wie Funktionsdefinitionen geschrieben, befinden sich aber innerhalb der Klasse. Auch sie geben in der Regel einen Wert zurück, so wie es Funktionen tun.

Wenn die Methode eines Objekts aufgerufen wird, muss sie oft auf in den Objekteigenschaften gespeicherte Werte zugreifen oder diese aktualisieren.

PHP bietet die spezielle Variable `$this`, mit deren Hilfe Methoden mit Werten arbeiten können, die in den Eigenschaften dieses Objekts gespeichert sind.

EIGENSCHAFTSWERTE BEIM ERSTELLEN VON OBJEKten ZUWEISEN

Siehe Seiten 160 bis 163

Anstatt ein Objekt zu erstellen und dann die Werte für jede seiner Eigenschaften einzeln festzulegen, können Sie ein Objekt erstellen und den Eigenschaften in einer einzigen Codezeile Werte zuweisen. Zu diesem Zweck wird der Klasse eine sogenannte **Konstruktorfunktion** hinzugefügt.

In PHP 8 kann eine Konstruktorfunktion auch die Eigenschaften eines Objekts definieren (sodass Sie sie erst definieren müssen, wenn Sie ihre Werte in einer Konstruktorfunktion festlegen).

STEUERN, WELCHER CODE AUF EIGENSCHAFTEN ZUGREIFEN KANN

Siehe Seite 164 bis 165

Manchmal sollen PHP-Seiten nicht direkt auf die Eigenschaften eines Objekts zugreifen oder sie aktualisieren. Stattdessen können Sie Methoden erstellen, die die in diesen Eigenschaften gespeicherten Werte abrufen oder aktualisieren.

Sie können zum Beispiel die Eigenschaft `balance` eines Kontoobjekts ausblenden und dann die Methoden `getBalance()`, `deposit()` und `withdraw()` verwenden, um mit dem Wert zu arbeiten, der in der Eigenschaft `balance` gespeichert ist.

KLASSEN: VORLAGEN FÜR OBJEKTE

Beim Erstellen einer **Klassendefinition** werden die Eigenschaften und Methoden eines Objekts in geschweiften Klammern gespeichert.

Eine Klassendefinition verwendet:

- das Schlüsselwort `class`.
- einen Namen, der die Art des Objekts beschreibt, das sie erzeugt. Der Name sollte in UpperCamelCase-Schreibweise sein: Dabei beginnt der erste Buchstabe jedes Wortes mit einem Großbuchstaben (keine Unterstriche verwenden).
- ein Paar geschweifte Klammern, um einen Code-block zu erstellen. Die geschweiften Klammern zeigen an, wo die Klasse beginnt/endet. Jede Klammer beginnt in einer neuen Zeile.
- **HINWEIS:** Nach der schließenden geschweiften Klammer, die die Klassendefinition abschließt, steht kein Semikolon.

Innerhalb der geschweiften Klammern der Klasse werden die Eigenschaften für diese Objektart aufgelistet:

- ein Schlüsselwort für die Sichtbarkeit (siehe S. 164). Die folgenden Eigenschaften verwenden das Schlüsselwort `public`.
- der Datentyp, den die Eigenschaft enthalten soll (Dies wurde in PHP 7.4 hinzugefügt und ist optional.)
- der Name der Eigenschaft, beginnend mit einem \$-Symbol

Die Methoden verwenden dieselbe Syntax wie Funktionsdefinitionen, jedoch wird ihnen das Schlüsselwort `visibility` vorangestellt (siehe S. 164); die folgenden Methoden verwenden `public`.

```
class Account
{
    EIGEN-
    SCHAFTE
    [
        public int      $number;
        public string   $type;
        public float    $balance;

    METHODEN
    [
        public function deposit(float $amount): float
        {
            // Code zum Einzahlen von Geld
        }
        public function withdraw(float $amount): float
        {
            // Code zum Abheben von Geld
        }
    ]
}
```

OBJEKT MITHILFE EINER KLASSE ERSTELLEN

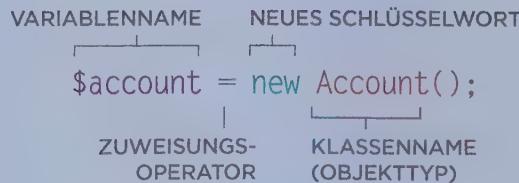
Um ein Objekt zu erstellen, verwenden Sie das Schlüsselwort new, gefolgt vom Klassennamen und einem Paar Klammern.

Um das Objekt zu erstellen, verwenden Sie:

- das Schlüsselwort new.
- den Namen der Klasse, die als Vorlage für das Objekt dienen soll.
- Klammern. Diese können Parameternamen enthalten (genau wie die eingeklammerten Parameter einer Funktion). Sie übergeben Daten an das Objekt, wenn es erstellt wird.

Wie Sie auf S. 150 gesehen haben, wird ein Verweis auf ein Objekt oft in einer Variablen gespeichert, damit es vom restlichen Code einer PHP-Seite verwendet werden kann. Dazu

- legen Sie eine Variable an, die das Objekt enthält; ihr Name sollte den Typ des Objekts beschreiben, das sie enthält.
- fügen Sie den Zuweisungsoperator = hinzu.
- erstellen Sie das Objekt, wie auf der linken Seite beschrieben.



Im obigen Beispiel soll die Variable \$account einen Verweis auf ein Objekt speichern, das mit der Klasse Account erstellt wurde (siehe linke Seite).

Es gibt drei Eigenschaften: `$number`, `$type` und `$balance`. Keine dieser Eigenschaften hat bisher einen Wert. Wie Sie ihnen Werte zuweisen, erfahren Sie auf der nächsten Seite.

Das Objekt verfügt auch automatisch über die beiden Methoden, die in der Klassendefinition enthalten sind.

Um ein zweites Objekt zu erstellen, das ein anderes Konto repräsentiert, gehen Sie genauso vor wie oben gezeigt, aber mit einem anderen Variablennamen (sonst würde das zweite Objekt das erste überschreiben).

Die Klassendefinition muss in jeder Seite enthalten sein, die ein Objekt mit dieser Klasse erstellt. Wenn eine Klassendefinition von mehr als einer Seite verwendet wird, wird sie in einer separaten Datei gespeichert, die dann in beide Seiten aufgenommen werden kann. Diese Datei erhält denselben Namen wie die Klasse (z.B. `Account.php`).

AUF EIGENSCHAFTEN ZUGREIFEN UND SIE AKTUALISIEREN

Der Zugriff auf die Eigenschaften eines Objekts und deren Aktualisierung eines Objekts entspricht der Verwendung von Variablen. Wenn ein Objekt in einer Variablen gespeichert ist, geben Sie zuerst den Variablenamen an und geben dann mit dem Objektoperator die Eigenschaft an, mit der Sie arbeiten möchten.

AUF EIGENSCHAFTEN ZUGREIFEN

Um auf einen in einer Eigenschaft gespeicherten Wert zuzugreifen, verwenden Sie

- den Name der Variablen, die das Objekt enthält,
- den Objektoperator -> ohne Leerzeichen davor und danach und
- den Namen der Eigenschaft (beachten Sie, dass der Eigenschaftsname hier nicht mit dem Symbol \$ beginnt).

Der Objektoperator zeigt an, dass die Eigenschaft auf der rechten Seite des Operators zu dem Objekt gehört, das in der Variablen auf der linken Seite des Operators gespeichert ist.

Der folgende Code zeigt, wie der Wert des Kontostands auf der Seite angezeigt wird.

```
OBJEKT      EIGENSCHAFT  
echo $account->balance;  
          |  
          OBJEKT-  
          OPERATOR
```

Wenn der Datentyp einer Eigenschaft in der Klasse festgelegt wurde und Sie versuchen, auf die Eigenschaft zuzugreifen, bevor ihr ein Wert zugewiesen wurde, erzeugt der PHP-Interpreter einen Fehler, der die Ausführung der Seite stoppen kann.

EIGENSCHAFTEN EINSTELLEN UND AKTUALISIEREN

Um den in einer Eigenschaft gespeicherten Wert zu aktualisieren, verwenden Sie

- den Namen der Variablen, die das Objekt enthält,
- den Objektoperator -> ohne Leerzeichen davor und danach,
- den Name der Eigenschaft, die Sie aktualisieren möchten,
- den Zuweisungsoperator = und
- den neuen Wert (wenn der Wert eine Zeichenkette ist, wird er in Anführungszeichen gesetzt; Zahlen und boolesche Werte werden nicht in Anführungszeichen gesetzt).

Wenn Sie versuchen, den Wert einer Eigenschaft zu setzen, die nicht in der Klassendefinition enthalten ist, wird die Eigenschaft zu diesem einen Objekt hinzugefügt (nicht aber zu anderen Objekten, die mit der Klasse erstellt wurden).

OBJEKT	EIGENSCHAFT	WERT
\$account->number	= 20148896;	
\$account->type	= 'Checking';	
\$account->balance	= 1000.00;	
OBJEKT- OPERATOR	ZUWEISUNGS- OPERATOR	

Wie Sie auf S. 161 lernen, ist es möglich, in der Methode `__construct()` Standardwerte für Eigenschaften anzugeben. Dadurch wird sichergestellt, dass jede Eigenschaft einen Wert hat, wenn das Objekt mit der Klasse erstellt wird.

OBJEKTE UND EIGENSCHAFTEN VERWENDEN

PHP

section_a/c04/objects-and-properties.php

```
<?php  
class Customer  
{  
    public string $forename;  
    public string $surname;  
    public string $email;  
    public string $password;  
}  
  
class Account  
{  
    public int    $number;  
    public string $type;  
    public float  $balance;  
}  
  
③ $customer = new Customer();  
④ $account  = new Account();  
⑤ $customer->email  = 'ivy@eg.link';  
⑥ $account->balance = 1000.00;  
?  
⑦ <?php include 'includes/header.php'; ?>  
⑧ <p>Email: <?= $customer->email ?></p>  
⑨ <p>Balance: $<?= $account->balance ?></p>  
⑩ <?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Die Klasse Customer und ihre Eigenschaften werden definiert.
2. Die Klasse Account und ihre Eigenschaften werden definiert.
3. Es wird eine Instanz der Klasse Customer erstellt und dieses Objekt in der Variablen \$customer gespeichert.
4. Es wird eine Instanz der Klasse Account erstellt und dieses Objekt in der Variablen \$account gespeichert.
5. Die Eigenschaft email des Objekts Customer erhält einen Wert.
6. Die Eigenschaft balance des Objekts Account wird mit einem Wert versehen.
7. Eine Include-Datei fügt der Seite eine Kopfzeile hinzu.
8. Die beiden soeben eingesetzten Eigenschaften werden angezeigt.
9. Eine Include-Datei fügt die HTML-Tags hinzu, die zum Schließen der Seite benötigt werden.
Probieren Sie es: Nach Schritt 5 legen Sie den Vor- und Nachnamen des Kunden im Objekt Customer fest.
Zeigen Sie dann in Schritt 8 den Namen des Kunden vor der E-Mail-Adresse an.

METHODEN DEFINIEREN UND AUFRUFEN

Eine Methode ist eine Funktion, die innerhalb einer Klassendefinition geschrieben wird. Um eine Methode aufzurufen, verwenden Sie den Namen der Variablen, die das Objekt enthält, den Objektoperator und dann den Methodennamen.

EINE METHODE DEFINIEREN

Um eine Methode zu einer Klasse hinzuzufügen, verwenden Sie ein Sichtbarkeitsschlüsselwort (siehe S.164 – unten wird `public` verwendet), gefolgt von einer Funktionsdefinition. Wenn eine Methode auf eine Eigenschaft des Objekts zugreifen oder diese aktualisieren muss, verwenden Sie

- die spezielle Variable `$this` (bekannt als **Pseudo-variable**), um anzugeben, dass Sie auf eine Eigenschaft dieses Objekts zugreifen möchten,
- den Objektoperator `->` und
- den Namen der Eigenschaft, auf die Sie zugreifen möchten.

Die unten stehende Methode `deposit()` hat den Parameter `$amount`. Wenn die Methode aufgerufen wird, wird der für `$amount` verwendete Wert zum Wert in der Eigenschaft `balance` addiert und der neue Wert in `balance` zurückgegeben.

```
class Account
{
    public int    $number;
    public string $type;
    public float  $balance;

    public function deposit($amount)
    {
        $this->balance += $amount;
        return $this->balance;
    }
}
```

EINE METHODE AUFRUFEN

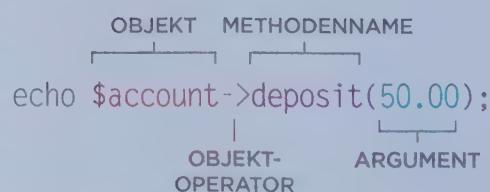
Um eine Methode aufzurufen, verwenden Sie

- den Namen der Variablen, die das Objekt enthält,
- den Objektoperator `->`,
- den Namen der Methode und
- Argumente für die Parameter der Methode.

Im folgenden Beispiel werden 50\$ auf das Konto eingezahlt.

Die Methode `deposit()` (in der linken Spalte) fügt diesen Betrag zum Wert der Eigenschaft `balance` hinzu und gibt dann den neuen Saldo zurück.

Mit dem Befehl `echo` wird der neuen Saldo auf die Seite geschrieben.



METHODEN VON OBJEKten VERWENDEN

PHP

```
<?php  
class Account  
{  
    public int $number;  
    public string $type;  
    public float $balance;  
  
    public function deposit(float $amount): float  
    {  
        $this->balance += $amount;  
        return $this->balance;  
    }  
  
    public function withdraw(float $amount): float  
    {  
        $this->balance -= $amount;  
        return $this->balance;  
    }  
  
}  
  
$account = new Account();  
$account->balance = 100.00;  
?  
<?php include 'includes/header.php'; ?>  
8 <p>$<?= $account->deposit(50.00) ?></p>  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Definieren Sie die Klasse Account und ihre Eigenschaften (wie auf S. 157).
2. Fügen Sie die Methode deposit() hinzu. Der Parameter \$amount ist der Betrag, der zum Guthaben hinzugefügt werden soll.
3. Der an die Funktion übergebene Betrag wird zu dem Wert addiert, der in der Eigenschaft balance gespeichert ist:
 - \$this->balance ruft die balance-Eigenschaft dieses Objekts ab.
 - += fügt den Wert in \$amount zum Saldo hinzu.
4. Der neue Wert, der in der Eigenschaft balance gespeichert ist, wird zurückgegeben.
5. withdraw() macht dasselbe wie deposit(), zieht aber einen Betrag vom Saldo ab.
6. Ein Objekt wird mit der Klasse Account erstellt und in der Variablen \$account gespeichert.
7. Die Eigenschaft balance des Objekts wird auf 100.00 gesetzt.
8. Die Methode deposit() wird aufgerufen und dem Konto werden 50,00\$ hinzugefügt. Sie gibt den aktualisierten Saldo zurück, und dieser Wert wird mit der Abkürzung echo auf die Seite geschrieben.

Probieren Sie es: Nach Schritt 8
heben Sie mit withdraw() 75\$ ab.

KONSTRUKTOR-METHODE

Die Methode `__construct()` wird als **Konstruktor** bezeichnet.

Wenn Sie die Methode `__construct()` zu einer Klassendefinition hinzufügen (ihr Name muss mit **zwei** Unterstrichen beginnen), werden die Anweisungen innerhalb dieser Methode automatisch ausgeführt, wenn die Klasse zur Erstellung eines Objekts verwendet wird.

Eine `__construct()`-Methode kann verwendet werden, um in einer einzigen Codezeile ein Objekt zu erstellen und seinen Eigenschaften Werte hinzuzufügen, anstatt das Objekt zu erstellen und dann jede Eigenschaft einzeln mit einer separaten Anweisung zu setzen (wie auf S. 156 bis 157 geschehen).

Im Folgenden wird ein Objekt mit der Klasse `Account` erstellt und in der Variablen `$account` gespeichert. Wenn das Objekt erstellt wird, sucht der PHP-Interpreter in der Klasse nach der Methode `__construct()`.

Die Argumente in den Klammern hinter dem Klassenname werden an die Methode `__construct()` übergeben (wie auf der rechten Seite gezeigt). Innerhalb der `__construct()`-Methode werden mit diesen Werten die Objekteigenschaften festgelegt.

VARIABLE	KLASSENNAME	WERTE, DIE ZUR OBJEKTERSTELLUNG VERWENDET WERDEN
<code>\$account</code>	<code>= new Account(20148896,</code>	<code>'Checking', 1000.00);</code>

HINWEIS: Beginnen Sie den Namen Ihrer eigenen Funktionen nicht mit zwei Unterstrichen; diese Namenskonvention sollte nur für die sogenannten **magischen Methoden** von PHP verwendet werden.

Magische Methoden werden automatisch vom PHP-Interpreter aufgerufen; Sie müssen sie nicht in Ihrem eigenen Code aufrufen.

Im Folgenden hat die Methode `__construct()` der Klasse `Account` drei Parameter: `$type`, `$number` und `$balance`, die den Eigenschaften entsprechen. Die drei Anweisungen in der Methode `__construct()` nehmen die Werte in den Parametern und verwenden diese, um die Objekteigenschaften festzulegen.

Wie Sie auf Seite 158 gesehen haben, können Sie mit der Pseudovariablen `$this` auf die Eigenschaften **dieses** Objekts zugreifen oder sie aktualisieren.

Wenn ein Objekt mithilfe des Codes auf der linken Seite erstellt wurde, würde die unten gezeigte `__construct()`-Methode automatisch ausgeführt, und die folgenden Werte würden zugewiesen:

- Parameter `$number`: 20148896
- Parameter `$type`: 'Checking'
- Parameter `$balance`: 100.00

Im Beispiel auf der nächsten Seite sehen Sie, wie Sie für jede dieser Eigenschaften Standardwerte festlegen können.

```
class Account
{
    public int      $number;
    public string   $type;
    public float    $balance;

    public function __construct($number, $type, $balance)
    {
        $this->number = $number;
        $this->type   = $type;
        $this->balance = $balance;
    }

    function deposit($amount) {...}
    function withdraw($amount) {...}
    function getBalance() {...}
}
```

PHP 8 bietet einen einfacheren Weg, Klassendefinitionen zu schreiben, indem Sie die Eigenschaften einer Klasse innerhalb der Klammern der `__construct()` Methode deklarieren.

Wenn ein Objekt mit der Klasse erstellt wird, werden die dem Konstruktor übergebenen Argumente automatisch als Werte für diese Eigenschaften zugewiesen. Dies wird als **Constructor Property Promotion** bezeichnet.

Wenn eine Eigenschaft optional ist, kann ein Standardwert angegeben werden (siehe die Eigenschaft `$balance` auf der rechten Seite), und dieser Wert wird verwendet, wenn kein Argument angegeben wird.

```
class Account
{
    public function __construct(
        public int      $number,
        public string   $type,
        public float    $balance =
        0.00,
    ) {}

    function deposit($amount) {...}
    function withdraw($amount) {...}
    function getBalance() {...}
}
```

KONSTRUKTOREN MIT EINER KLASSE VERWENDEN

1. Die PHP-Seite beginnt mit der Aktivierung von strikten Typen, da Typdeklarationen zu den Methoden hinzugefügt wurden (siehe S. 126–127).
2. Der Klassename und seine Eigenschaften werden definiert.

3. Die Methode `__construct()` von der vorherigen Seite wird hinzugefügt, um die Werte der Eigenschaften festzulegen. Den Parametern werden Argumenttyp-Deklarationen hinzugefügt. Wenn bei der Erstellung des Objekts kein Saldo angegeben wird, wird ein Standardwert von 0,00 verwendet.

4. Die Methoden `deposit()` und `withdraw()` aktualisieren den in der Eigenschaft `balance` gespeicherten Wert. Sie erhalten als Argument und Rückgabetyp die Deklaration `float`. (Wenn der Datentyp `float` ist, kann `int` verwendet werden, ohne einen Fehler zu verursachen).

section_a/c04/constructor-methods.php

PHP

```
<?php
① declare(strict_types = 1);
class Account
{
    ②   public int      $number;
        public string $type;
        public float   $balance;

    ③   public function __construct(int $number, string $type, float $balance = 0.00)
    {
        $this->number      = $number;
        $this->type        = $type;
        $this->balance     = $balance;
    }

    ④   public function deposit(float $amount): float
    {
        $this->balance += $amount;
        return $this->balance;
    }

    public function withdraw(float $amount): float
    {
        $this->balance -= $amount;
        return $this->balance;
    }
}
```

```

⑤ [ $checking = new Account(43161176, 'Checking', 32.00);
  $savings  = new Account(20148896, 'Savings', 756.00);
  ?>

<?php include 'includes/header.php'; ?>
<h2>Account Balances</h2>
<table>
  <tr>
    <th>Date</th>
    <th><?= $checking->type ?></th>
    <th><?= $savings->type ?></th>
  </tr>
  <tr>
    <td>23 June</td>
    <td>$<?= $checking->balance ?></td>
    <td>$<?= $savings->balance ?></td>
  </tr>
  <tr>
    <td>24 June</td>
    <td>$<?= $checking->deposit(12.00) ?></td>
    <td>$<?= $savings->withdraw(100.00) ?></td>
  </tr>
  <tr>
    <td>25 June</td>
    <td>$<?= $checking->withdraw(5.00) ?></td>
    <td>$<?= $savings->deposit(300.00) ?></td>
  </tr>
</table>
<?php include 'includes/footer.php'; ?>

```

ERGEBNIS

ACCOUNT BALANCES		
DATE	CHECKING	SAVINGS
23 June	\$32	\$756
24 June	\$44	\$656
25 June	\$34	\$966

5. Es werden zwei Objekte für ein Girokonto und ein Sparkonto erstellt.

Der Konstruktor ordnet den Eigenschaften der beiden Objekte die Werte in den Klammern zu.

6. Eine HTML-Tabelle wird in der Seite erstellt. Die erste Zeile enthält Überschriften, die die Eigenschaft type der beiden Objekte verwenden. Um auf eine Eigenschaft zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das Objekt enthält
- den Objektorientator
- den Namen der Eigenschaft

7. Die nächste Tabellenzeile zeigt die Eigenschaft balance der Objekte.

8. In der dritten Tabellenzeile wird der Saldo jedes Kontos durch den Aufruf der Methoden deposit() oder withdraw() aktualisiert.

Diese Methoden geben den neuen Wert der balance-Eigenschaft zurück, und dieser wird auf der Seite ausgegeben. Um eine Methode aufzurufen, verwenden Sie:

- den Namen der Variablen, die das Objekt enthält
- den Objektorientator
- den Namen der Methode mit ihren Argumenten in Klammern

9. In der vierten Zeile der Tabelle wird der vorherige Schritt mit anderen Werten wiederholt.

Probieren Sie es: Erstellen Sie in Schritt 6 ein Objekt für ein hochverzinsliches Konto. Fügen Sie in den Schritten 7–9 Zeilen hinzu, um die Aktualisierung des Saldos zu zeigen.

SICHTBARKEIT VON EIGENSCHAFTEN UND METHODEN

Sie können verhindern, dass Code außerhalb eines Objekts die in den Objekteigenschaften gespeicherten Werte abrufen oder setzen kann. Genauso können Sie verhindern, dass der Code außerhalb eines Objekts dessen Methoden aufruft.

Die Eigenschaften und Methoden einer Klasse werden als **Mitglieder** einer Klasse bezeichnet. Sie können festlegen, ob der Code außerhalb eines Objekts, das mit dieser Klasse erstellt wurde, gültig ist:

- Zugriff auf den in einer Eigenschaft gespeicherten Wert oder dessen Aktualisierung
- Aufruf einer Methode

Dies geschieht durch die Einstellung der **Sichtbarkeit** bei der Deklaration einer Eigenschaft oder der Definition einer Methode.

Bislang wurde in diesem Kapitel allen Eigenschafts- und Methodennamen das Wort `public` vorangestellt. Das bedeutet, dass jeder andere Code mit den Eigenschaften und Methoden des Objekts arbeiten kann.

In manchen Fällen darf nur der Code innerhalb des Objekts auf Eigenschaften zugreifen oder diese aktualisieren oder die Methoden des Objekts aufrufen. Zu diesem Zweck ändern Sie das Wort `public` in `protected`.

Die Klasse `Account` hat beispielsweise die Eigenschaft `balance`. Wenn sie mit dem Schlüsselwort `public visibility` deklariert wird, kann jeder Code, der ein Objekt mit dieser Klasse erstellt, den Wert dieser Eigenschaft abrufen oder aktualisieren.

Um zu verhindern, dass anderer Code den in der Eigenschaft `balance` gespeicherten Wert aktualisiert, kann ihre Sichtbarkeit auf `protected` gesetzt werden.

Wenn Sie versuchen, mit Code außerhalb der Klasse auf eine geschützte Eigenschaft zuzugreifen, erzeugt der PHP-Interpreter einen Fehler.

Wenn Code außerhalb des Objekts den in einer `protected`-Eigenschaft gespeicherten Wert abrufen muss, fügen Sie der Klasse eine Methode hinzu, die ihren Wert zurückgibt. Diese Methode wird als **Getter** bezeichnet (weil sie einen Wert abruft).

Im Beispiel rechts wird der Klasse `Account` die neue Methode `getBalance()` hinzugefügt, die den in der Eigenschaft `$balance` gespeicherten Wert zurückgeben soll.

Um den in einer geschützten Eigenschaft gespeicherten Wert zu aktualisieren, fügen Sie der Klasse eine Methode hinzu, die ihren Wert aktualisiert. Diese Methode wird als **Setter** bezeichnet (weil sie einen Wert festlegt oder setzt).

Die Methoden `deposit()` und `withdraw()` in der Klasse `Account` werden bereits zur Aktualisierung des in der Eigenschaft `$balance` gespeicherten Werts verwendet.

Diese Änderungen gewährleisten, dass der Saldo nur durch die Methoden `deposit()` oder `withdraw()` aktualisiert wird. Er kann durch keinen anderen Code aktualisiert werden.

Wenn Sie die Sichtbarkeit einer Eigenschaft oder Methode in der Klassendefinition nicht angeben, ist sie standardmäßig `public`, aber die explizite Angabe, ob die Eigenschaft oder Methode `public` oder `protected` ist, gilt als gute Praxis und macht Ihren Code leichter verständlich.

Es gibt noch eine weitere Einstellung für die Sichtbarkeit namens `private`, die in fortgeschrittenem objektorientiertem Code verwendet wird. Dies würde den Rahmen eines Einsteigerbuchs sprengen. Diese Einstellungen werden auch **Zugriffsmodifikatoren** genannt.

GETTER UND SETTER VERWENDEN

PHP

section_a/c04/getters-and-setters.php

```
<?php
declare(strict_types = 1);

class Account {
    public int $number;
    public string $type;
    protected float $balance;

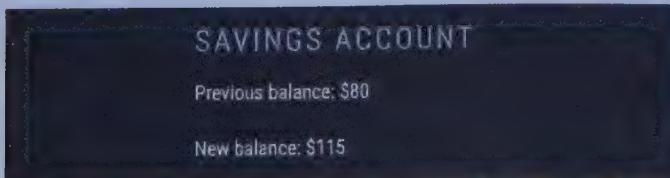
①    public function __construct() {...}
②    public function deposit() {...}
    public function withdraw() {...}

    public function getBalance(): float
    {
        ③        return $this->balance;
    }

④    $account = new Account(20148896, 'Savings', 80.00);
    ?>

    <?php include 'includes/header.php'; ?>
⑤    <h2><?= $account->type ?> Account</h2>
⑥    <p>Previous balance: $<?= $account->getBalance() ?></p>
⑦    <p>New balance: $<?= $account->deposit(35.00) ?></p>
    <?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Die Eigenschaft `balance` war bisher `public` und wird nun in `protected` geändert, damit sie außerhalb der Klasse nicht sichtbar ist.

2. Die bestehenden Methoden `deposit()` und `withdraw()` dienen als Setter, um den Saldo zu aktualisieren (der Code für diese Methoden ist derselbe wie im vorherigen Beispiel).

3. Der Getter `getBalance()` wird der Klasse hinzugefügt, um den Wert der geschützten Eigenschaft `balance` zu ermitteln, falls er angezeigt werden muss.

4. Es wird ein Objekt mit der Klasse `Account` erstellt. Es wird in der Variablen `$account` gespeichert.

5. Die Art des Kontos wird angezeigt. Da diese Eigenschaft `public` ist, kann auf sie direkt zugegriffen werden.

6. Die Methode `getBalance()` wird aufgerufen, um den Wert in der Eigenschaft `$balance` anzuzeigen.

7. Die Methode `deposit()` wird aufgerufen. Sie fügt 35\$ zur Eigenschaft `$balance` hinzu. Diese Methode gibt auch den neuen Saldo zurück, sodass er auf der Seite ausgegeben werden kann.

Probieren Sie es: Verwenden Sie nach Schritt 6 die Methode `withdraw()`, um 50\$ vom Konto abzuheben.

EIN ARRAY IN EINER OBJEKTEIGENSCHAFT SPEICHERN

Eine Objekteigenschaft kann ein Array speichern. Auf einzelne Elemente des Arrays kann dann mit der Array-Syntax zugegriffen werden.

Alle Objekteigenschaften, die Sie bisher kennengelernt haben, speichern skalare Datentypen (Strings, Zahlen und boolesche Werte). Die Objekteigenschaft kann auch einen zusammengesetzten Datentyp wie z.B. ein Array speichern. Im Folgenden wird ein Account-Objekt in der Variablen \$account gespeichert, und seine Eigenschaft number wird festgelegt.

Der der Eigenschaft number zugewiesene Wert ist ein assoziatives Array mit zwei separaten Werten:

- einer Kontonummer
- einer Bankleitzahl

OBJEKT	EIGENSCHAFT	WERT IST EIN ARRAY
\$account->number	=	['account_number' => 12345678, 'routing_number' => 987654321,];

Um auf einen Wert zuzugreifen, der als Array in der number-Eigenschaft enthalten ist, verwenden Sie:

- Name der Variablen, die das Objekt enthält
- Objektorientator
- Name der Eigenschaft, die das Array enthält
- Schlüssel des Elements im Array, auf das Sie zugreifen möchten

Im Folgenden werden die Kontonummer und die Bankleitzahl, die als assoziatives Array in der number-Eigenschaft des Account-Objekts gespeichert wurden, unter Verwendung ihrer Schlüsselnamen abgerufen und mit dem echo-Befehl auf der Seite ausgegeben. (Würde es sich um ein indiziertes Array handeln, wäre der Schlüssel die Indexnummer des Elements, auf das Sie zugreifen möchten).

OBJEKT	EIGENSCHAFT	SCHLÜSSEL
echo \$account->number['account_number'];		
echo \$account->number['routing_number'];		

EIN ARRAY IN EINER OBJEKTEIGENSCHAFT VERWENDEN

PHP

section_a/c04/array-in-object.php

```
<?php
declare(strict_types = 1);

① class Account {...}
// Wie S. 165, aber number ist ein Array (kein int)

//Array für die Eigenschaft erstellen
② $numbers = ['account_number' => 12345678,
              'routing_number' => 987654321,];

//Klasseninstanz und Eigenschaften
③ $account = new Account($numbers, 'Savings', 10.00);
?>
<?php include 'includes/header.php'; ?>
④ <h2><?= $account->type ?> account</h2>
⑤ Account <?= $account->number['account_number']
⑥ ?><br>
Routing <?= $account->number['routing_number'] ?>
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



In diesem Beispiel wird ein Objekt für ein Konto erstellt. Die Kontonummer und die Bankleitzahl werden in der Eigenschaft \$number gespeichert.

1. Die Klasse Account entspricht der im vorherigen Beispiel auf S.165, aber die Argumenttypdeklaration für den Parameter \$number in der Methode __construct() signalisiert, dass der Wert ein Array sein wird.

2. Die Variable \$number wird deklariert. Sie enthält ein assoziatives Array mit zwei Schlüsseln:

- account_number
- routing_number

3. Es wird ein Objekt mit der Klasse Account erstellt. Das erste Argument ist die in Schritt 2 erstellte Variable \$numbers. Damit wird das Array der Eigenschaft \$number des Objekts zugewiesen.

4. Der Typ des Kontos wird auf der Seite ausgegeben.

5. Die Kontonummer wird angezeigt.

6. Die Bankleitzahl wird angezeigt.

Probieren Sie es: Ändern Sie in Schritt 2 die Kontonummer (account_number) und die Bankleitzahl (routing_number).

EIN OBJEKT IN EINER OBJEKTEIGENSCHAFT SPEICHERN

Eine Objekteigenschaft kann ein anderes Objekt speichern. Sie können dann auf einzelne Eigenschaften beider Objekte zugreifen oder diese aktualisieren und ihre Methoden aufrufen.

Auf der vorherigen Seite haben Sie gesehen, dass eine Objekteigenschaft ein Array speichern kann. Sie können auch ein anderes Objekt in der Objekteigenschaft speichern.

Im Folgenden ist der Wert, der der Eigenschaft `$number` zugewiesen wurde, ein neues Objekt, das mit der Klasse `AccountNumber` erstellt wurde (auf der rechten Seite dargestellt).

Die Klasse `AccountNumber` ist eine Vorlage für ein Objekt, das Kontonummern repräsentiert. Sie hat zwei Eigenschaften:

- `$accountNumber` enthält die Kontonummer
- `$routingNumber` enthält die Bankleitzahl



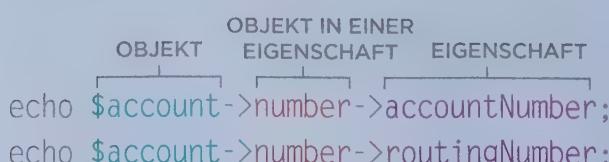
Um auf eine Eigenschaft oder Methode des Objekts zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das `Account`-Objekt enthält
- den Objektorientator
- den Namen der Eigenschaft, die Kontonummern speichert
- den Objektorientator (für den Zugriff auf dieses Objekt)
- Eigenschaft oder Methode, die Sie verwenden möchten

Im Folgenden enthält die Variable `$account` das Objekt, das das Bankkonto repräsentiert.

Die Eigenschaft `$number` speichert ein zweites Objekt, das mit der Klasse `AccountNumber` erstellt wurde.

Seine Eigenschaften `$accountNumber` und `$routingNumber` werden mit dem Befehl `echo` ausgegeben.



EIN OBJEKT ALS OBJEKTEIGENSCHAFT VERWENDEN

PHP

section_a/c04/object-in-object.php

```
<?php
declare(strict_types = 1);
① class Account {...}
// Wie S. 165, aber der Datentyp der number-Eigenschaft
// ist der Klassename AccountNumber

② class AccountNumber
{
    public int $accountNumber;
    public int $routingNumber;

    public function __construct(int $accountNumber,
                                int $routingNumber)
    {
        $this->accountNumber = $accountNumber;
        $this->routingNumber = $routingNumber;
    }
}

④ $numbers = new AccountNumber(12345678, 987654321);
⑤ $account = new Account($numbers, 'Savings', 10.00);
?>
<?php include 'includes/header.php';?>
⑥ <h2><?= $account->type ?> Account</h2>
⑦ Account <?= $account->number->accountNumber ?><br>
⑧ Routing <?= $account->number->routingNumber ?>
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS

SAVINGS ACCOUNT

Account 12345678
Routing 987654321

1. Die Klasse Account entspricht dem Beispiel auf S. 165, außer dass die Argumenttypdeklaration für den Parameter \$number in der Methode __construct() zeigt, dass der Wert für diesen Parameter ein Objekt sein sollte, das mit der Klasse AccountNumber erstellt wurde.

2. Es wird eine Klassendefinition für die Klasse AccountNumber hinzugefügt. Sie hat zwei public-Eigenschaften:

- \$accountNumber
- \$routingNumber

3. Mit einer Konstruktormethode werden diesen Eigenschaften Werte zugewiesen, wenn ein Objekt mit dieser Klasse erstellt wird.

4. Ein Objekt wird mit der Klasse AccountNumber erstellt. Es wird in der Variablen \$numbers gespeichert.

5. Es wird ein Objekt für ein Konto mit der Klasse Account erstellt. Das erste Argument ist die Variable, die das Objekt für die Kontonummern enthält.

6. Die Art (type) des Kontos wird in die Überschrift der Seite geschrieben.

7. Die Kontonummer wird angezeigt.

8. Die Bankleitzahl wird angezeigt.

VORTEILE DER VERWENDUNG VON OBJEKTEN

Objekte helfen Ihnen bei der Organisation Ihres Codes, ersparen die Wiederholung desselben Codes auf verschiedenen Seiten, machen die Wartung einfacher und erleichtern die gemeinsame Nutzung.

VERBESSERTE ORGANISATION

Wenn eine PHP-Seite Hunderte von Codezeilen enthält, kann es schwierig sein, herauszufinden, was jede einzelne Codezeile bewirkt.

Wenn Sie die Variablen und Funktionen, die für ein Konzept, z.B. einen Kunden oder sein Konto, verwendet werden, in einer Klasse gruppieren, können Sie den gesamten zugehörigen Code an einem Ort halten.

Werden Objekte mithilfe einer Klasse erstellt, können Sie in der Klassendefinition nachsehen,

- welche Daten in den Eigenschaften verfügbar sind,
- welche Aufgaben mit den Methoden ausgeführt werden können.

Wie Sie im letzten Beispiel dieses Kapitels auf der nächsten Seite sehen werden, werden Klassendefinitionen oft in separaten Dateien gespeichert (die als **Klassendateien** bezeichnet werden). Dies erleichtert das Auffinden des Codes für eine Klasse.

BESSERE WIEDERVERWENDBARKEIT

Es kann mehrere Seiten einer Website geben, die dieselben Dinge repräsentieren müssen; zum Beispiel kann es mehrere Seiten für einen Kunden oder ein Mitglied einer Website geben.

Anstatt auf jeder Seite die Variablen-deklarationen (zur Speicherung der Kundendaten) und die Funktionsdefinitionen (zur Darstellung der Aufgaben, die die Kunden ausführen können) zu wiederholen, kann eine Klassen-definition als Vorlage verwendet werden, um ein Objekt zu erstellen.

Jede Seite, die einen Kunden repräsentieren soll, kann die Klassendatei in die Seite einbinden und ein Objekt erstellen, das diese Klassendefinition als Vorlage verwendet.

Bei der Programmierung beruft man sich manchmal auf den Grundsatz »**Don't Repeat Yourself**«, das DRY-Prinzip. Demzufolge sollten Sie, wenn Sie merken, dass Sie Code wiederholen, prüfen, ob nicht stattdessen eine Funktion oder Methode eines Objekts zur Ausführung der Aufgabe verwendet werden kann.

In der Programmierung bezieht man sich auch auf das **Prinzip einer einzigen Verantwortung (single responsibility principle)**, das besagt, dass jede Funktion oder Methode nur eine einzige Aufgabe (und nicht mehrere) erfüllen sollte. Dies trägt zur Optimierung der Code-Wiederverwendbarkeit bei und macht ihn leichter verständlich.

LEICHTER ZU PFLEGEN

Eine sorgfältige Code-Organisation und die Maximierung der Wiederverwendung von Code tragen zu dessen einfacher Wartbarkeit bei.

Ein Beispiel:

- Wenn Sie zusätzliche Informationen über die Kunden einer Website speichern müssen, können Sie der Klassendefinition eine Eigenschaft für die Kunden hinzufügen. Diese Informationen sind dann für jedes Kundenobjekt verfügbar.
- Wenn Sie die Art und Weise ändern müssen, wie ein Programm eine bestimmte Aufgabe ausführt (z.B. wie die Zinsen für ein Konto berechnet werden), brauchen Sie nur den Code in einer Klasse zu aktualisieren, und schon wird jedes mit dieser Klasse erstellte Objekt aktualisiert.

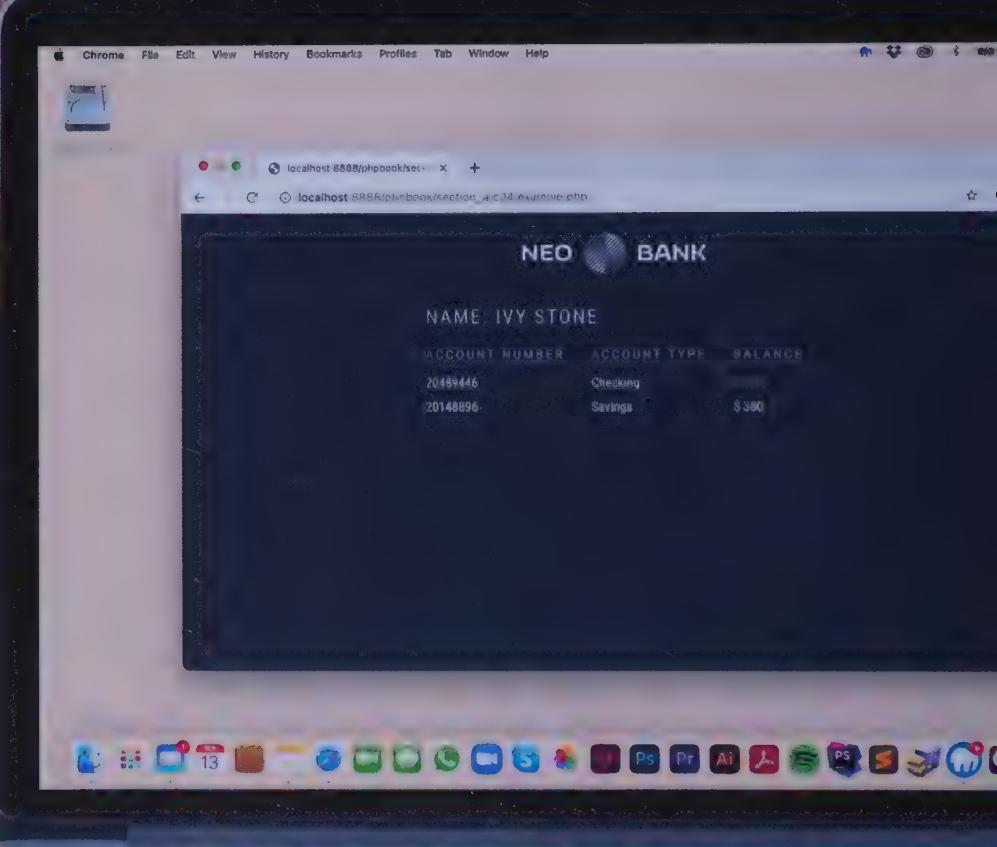
VEREINFACHTE FREIGABE VON CODE

Wenn Sie darüber nachdenken, wie eine Klasse geschrieben wird, brauchen Sie nicht zu wissen, wie sie alle ihre Aufgaben erfüllt, solange Sie ihren Namen, ihre Eigenschaften und Methoden kennen. Sie müssen nur Folgendes wissen:

- wie man ein Objekt mit dieser Klasse erstellt
- welche Daten Sie über die Eigenschaften erhalten können
- welche Aufgaben Sie mit den Methoden der Klasse durchführen können

Dies ist für Entwicklungsteams hilfreich, da verschiedene Teammitglieder für verschiedene Klassendefinitionen verantwortlich sein können.

Im nächsten Abschnitt des Buchs erfahren Sie, dass der PHP-Interpreter viele eingebaute Funktionen und Klassen bereithält, die Ihnen beim Erstellen von Webseiten helfen. Sie müssen nicht wissen, wie sie ihre Aufgaben erfüllen, Sie müssen nur wissen, wie man sie nutzt.



BEISPIEL

In diesem Beispiel werden Benutzerinformationen und Banksalden für einen Kunden mit mehreren Bankkonten angezeigt.

Dabei werden zwei Klassendefinitionen verwendet:

- Mit der Klasse `Customer` wird ein Objekt für einen Bankkunden erstellt.
- Mit der Klasse `Account` können Sie Objekte für verschiedene Konten eines Kunden erstellen.

Die Klassen werden in zwei separaten Dateien mit den Namen `Customer.php` und `Account.php` abgelegt, und diese Dateien werden in einem Ordner mit dem Namen `classes` gespeichert.

Jede Seite, die mit den Klassen ein Objekt erstellt, enthält die Klassendefinitionen, genauso wie die Kopf- und Fußzeilen der Seiten in jede Seite aufgenommen wurden.

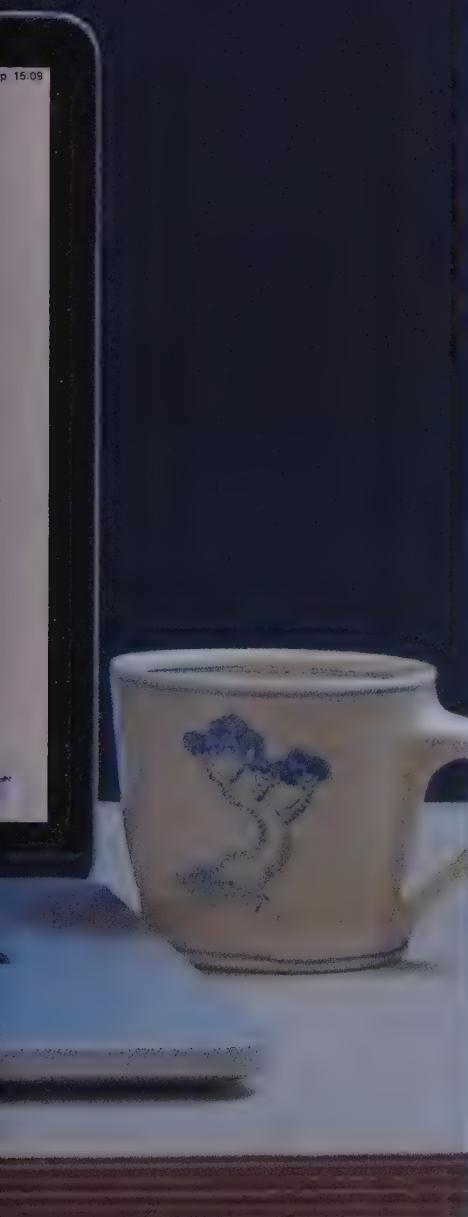
Auf der links abgebildeten Seite

- wird `Customer`-Objekt mithilfe der Klasse `Customer` erstellt,
- wird der Klasse `Customer` die neue Eigenschaft `$accounts` hinzugefügt,
- enthält die Eigenschaft `$accounts` ein Array,
- werden in diesem Array zwei `Account`-Objekte für die beiden Kontotypen des Kunden gespeichert (erstellt mit der neuen `Account`-Klasse)

Dies zeigt, dass Sie eine Objekthierarchie erstellen können, bei der ein Objekt ein anderes enthält.

Die Seite zeigt den Kundenamen an und durchläuft alle Konten des Kunden in einer `foreach`-Schleife. Innerhalb der Schleife werden die Kontonummer, der Kontotyp und der Saldo für jedes Konto angezeigt.

Eine bedingte Anweisung prüft, ob der Benutzer sein Konto überzogen hat, und wenn ja, wird der Saldo nicht weiß, sondern orange angezeigt.



BEISPIEL

section_a/c04/classes/Account.php

PHP

```
① <?php  
class Account {...} // Siehe Seite 165
```

section_a/c04/classes/Customer.php

PHP

```
<?php  
class Customer  
  
    public string $forename;  
    public string $surname;  
    public string $email;  
    private string $password;  
    public array $accounts;  
  
    function __construct(string $forename, string $surname, string $email,  
③                      string $password, array $accounts)  
    {  
        $this->forename = $forename;  
        $this->surname = $surname;  
        $this->email = $email;  
        $this->password = $password;  
        $this->accounts = $accounts;  
    }  
  
    function getFullName()  
    {  
        ⑤            return $this->forename . ' ' . $this->surname;  
    }  
;
```

Die beiden Klassendefinitionen werden in `Account.php` und `Customer.php` erstellt und in dem Ordner `classes` gespeichert. Die Klassen können mithilfe einer PHP-Include-Anweisung (siehe Schritt 5) in jede Seite eingebunden werden, auf der diese Objektart benötigt wird.

1. Die Klasse `Account` wurde auf S. 162–167 erstellt.

2. Die Klasse `Customer` baut auf der Klasse von S. 157 auf. Die neue Eigenschaft `$accounts` enthält ein Array von Objekten; jedes Objekt steht für eines der Konten des Kunden.

3 + 4. Die Eigenschaft `$accounts` wird der Konstruktormethode hinzugefügt.

5. Eine neue Methode gibt den vollständigen Namen des Kunden zurück.

6. Die Seite, auf der die Benutzerkonten angezeigt werden, enthält die Definitionen der Klassen `Account` und `Customer`, die zur Erstellung dieser Objekte erforderlich sind.

7. Ein indiziertes Array wird erstellt und in der Variablen `$accounts` gespeichert. Es enthält zwei Objekte, die mit der Klasse `Account` erstellt wurden. Jedes Objekt steht für eines der Bankkonten des Kunden.

```

<?php
⑥ [include 'classes/Account.php';
⑥ [include 'classes/Customer.php';

⑦ [$accounts = [new Account(20489446, 'Checking', -20),
    new Account(20148896, 'Savings', 380),];

⑧ $customer = new Customer('Ivy', 'Stone', 'ivy@eg.link', 'Jup!t3r2684', $accounts);
?>
<?php include 'includes/header.php'; ?>
⑨ <h2>Name: <b><?= $customer->getFullName() ?></b></h2>



```

8. Es wird ein neues Customer-Objekt für den Kunden erstellt und in der Variablen \$customer gespeichert. Das letzte Argument ist das Array der in Schritt 7 erstellten Konten.

9. Die neue Methode getFullName() des Customer-Objekts gibt den vollständigen Namen des Kunden zurück, der in der Überschrift angezeigt wird.

10. Eine foreach-Schleife durchläuft das Array, das in der Eigenschaft \$accounts des Customer-Objekts gespeichert ist. In der Schleife wird jedes Konto in der Variablen \$account gespeichert.

11. Die Kontonummer und der Kontotyp werden in die Seite geschrieben.

12. Eine if-Anweisung prüft, ob der Saldo 0 oder mehr beträgt.

13. Ist dies der Fall, wird ein <td>-Element mit der Klasse credit erstellt.

14. Wenn nicht, wird ein <td>-Element mit der Klasse overdrawn angelegt.

15. Der Saldo wird ausgeschrieben.

Probieren Sie es: Fügen Sie in Schritt 7 ein drittes Konto zu dem Array hinzu.

ZUSAMMENFASSUNG

OBJEKTE & KLASSEN

- Objekte fassen Variablen und Funktionen zusammen, die ein Objekt aus der Welt um uns herum repräsentieren.
- In einem Objekt werden die Variablen als Eigenschaften und die Funktionen als Methoden bezeichnet.
- Eine Klasse wird als Vorlage für die Erstellung von Objekten verwendet.
- Klassendefinitionen legen die Eigenschaften und Methoden fest, die jedes mit dieser Klasse erstellte Objekt haben soll.
- Die Methode `__construct()` wird ausgeführt, wenn ein Objekt erstellt wird. Mit ihr können Werte in Eigenschaften eingefügt werden.
- `$this` greift auf eine Eigenschaft oder Methode dieses Objekts zu.
- Eigenschaften können als `public` deklariert werden (auf sie kann von Code außerhalb des Objekts aus zugegriffen werden) oder als `protected` (sie können nur von Code innerhalb des Objekts verwendet werden).
- Klassen und Objekte helfen bei der Organisation, Wiederverwendung, Wartung und gemeinsamen Nutzung von Code.

IB

DYNAMISCHE WEBSEITEN

In diesem Teil lernen Sie, dynamische Webseiten mit PHP zu erstellen. Auf solchen Seiten angezeigte Inhalte können sich auch ohne manuelle Anpassung der Datei verändern.

In Teil A wurde die Syntax von PHP vorgestellt. Sie haben dort gelernt, wie:

- Variablen und Arrays Daten speichern
- Operatoren aus mehreren Informationen einen einzelnen Wert erzeugen
- Bedingungen und Schleifen festlegen, wann Code ausgeführt wird .
- Funktionen und Klassen zusammenhängende Anweisungen zusammenfassen

In diesem Buchteil lernen Sie, diese grundlegenden Konzepte auf die Erstellung dynamischer Webseiten anzuwenden. Im Grunde ist ein Computer eine Maschine, die programmiert wird, um:

- Daten (oder **Input**) zu akzeptieren
- diese Daten zu **verarbeiten** und Operationen damit durchzuführen
- anschließend eine **Ausgabe** zu erzeugen, die Anwender sehen oder hören können
- bei Bedarf Daten zur späteren Verwendung zu **speichern**

Die PHP-Seiten, die Sie in diesem Buchteil zu schreiben lernen, sind wie einfache Programme; sie können Eingaben aus einem Webbrowser entgegennehmen, diese Daten verarbeiten und auf dieser Grundlage dann eine individuell auf den Besucher zugeschnittene HTML-Seite ausgeben. Sie lernen:

- eine Reihe von PHP-Funktionen und -Klassen zu verwenden
- von Browsern gesendete Daten zu erfassen und zu verarbeiten
- mit Bildern und anderen Dateien umzugehen, die von den Anwendern hochgeladen werden können
- Daten über Website-Besucher mithilfe von Cookies und Sessions zu speichern
- mit Fehlern umzugehen und Probleme in Ihrem Code aufzuspüren und zu beheben

Zur erfolgreichen Lektüre dieses Abschnitts müssen Sie verstehen, wie der PHP-Interpreter Anfragen behandelt und wie er sie beantwortet.

Bei der Bearbeitung von Seitenanfragen befolgen Server Regeln, die in Protokollen und Codierungsverfahren festgelegt sind:

HTTP-ANFRAGEN UND -ANTWORTEN

Das **HyperText Transfer Protocol (HTTP)** besteht aus verschiedenen Regeln, die die Kommunikation zwischen Browsern und Servern steuern. Aus diesem Grund beginnen die URLs von Websites entweder mit `http://` oder `https://`. HTTP legt fest, welche Daten:

- Browser an einen Server senden, um eine Datei anzufordern
- Server an den Browser senden, wenn sie mit der angeforderten Datei antworten

CODIERUNGSVERFAHREN

Computer stellen Text-, Bild- und Audiodaten mithilfe von Binärimformationen dar, die sich aus einer Abfolge von Nullen und Einsen zusammensetzen.

Codierungsverfahren sind Regeln, mit denen Computer Dinge, die Sie als Mensch sehen und hören können, in jene Nullen und Einsen übersetzen, die sie selbst verarbeiten können. Wenn Sie nicht wissen, wie Sie PHP das gewünschte Codierungsverfahren mitteilen, werden die Daten möglicherweise nicht korrekt verarbeitet oder angezeigt.

Der PHP-Interpreter umfasst mehrere Hilfsmittel, die Ihnen bei der Erstellung dynamischer Webseiten helfen:

ARRAYS, FUNKTIONEN, KLASSEN

Der PHP-Interpreter bringt bereits mit:

- **Superglobale Arrays:** Diese Arrays werden bei jeder Dateianfrage erstellt.
- **Eingebaute Funktionen:** Sie führen häufig benötigte Aufgaben aus.
- **Eingebaute Klassen:** zur Erstellung von Objekten für häufig benötigte Dinge.

FEHLERMELDUNGEN

Der PHP-Interpreter gibt bei Problemen **Fehlermeldungen** aus. Wenn Sie diese Meldungen richtig zu deuten lernen, können Sie Probleme in Ihrem Code beheben.

EINSTELLUNGEN

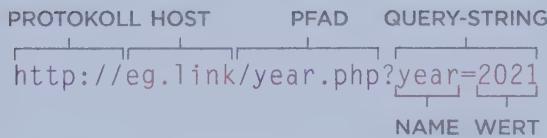
Wie viele andere Programme haben sowohl der PHP-Interpreter als auch der Webserver Einstellungen, die Sie anpassen können. Sie lernen, wie Sie die Einstellungen beider Programme mithilfe von Textdateien ändern können.

HTTP-ANFRAGEN UND ANTWORTEN

Das HyperText Transfer Protocol (HTTP) ist ein Regelwerk, in dem festgelegt ist, wie Browser Seiten anfordern und wie Server ihre Antwort formatieren sollen. Es ist hilfreich, zu verstehen, welche Daten während dieser Schritte gesendet werden.

Wenn ein Webbrowser eine PHP-Seite **anfordert**, steht in der Adressleiste des Browsers eine URL mit Angaben darüber, wo der Browser diese Seite finden kann. Jede URL enthält:

- ein **Protokoll** (für Webseiten ist dies HTTP oder HTTPS)
- einen **Host** (den Server, an den die Anfrage gesendet wird)
- einen **Pfad**, der die angeforderte Datei bezeichnet
- optional auch einen **Query-String** mit zusätzlichen Daten, die die Seite möglicherweise benötigt



Die in einem Query-String am Ende einer URL übermittelten Datenfragmente entsprechen jeweils einer Variablen; sie haben einen:

- **Namen**, der die übermittelten Daten beschreibt. Der Name bleibt bei jedem Aufruf der URL derselbe.
- **Wert** für dieses Datenelement. Der Wert kann sich bei jeder Anforderung der Seite ändern.

Bei der Anforderung einer Webseite sendet der Browser auch **HTTP-Request-Header** an den Server. Diese werden nicht (wie etwa die URL) im Hauptfenster des Browsers angezeigt, lassen sich aber in den Entwicklungs-Tools der meisten Browser einsehen (siehe unterer Screenshot).

Die Header enthalten Daten, die für den Server von Nutzen sein können, und auch sie ähneln dabei einer Variablen; sie haben einen:

- **Namen**, der die übermittelten Daten beschreibt. Der Name bleibt bei jedem Aufruf der URL derselbe.
 - **Wert** für dieses Datenelement.
- Die Header im unten stehenden Beispiel zeigen:
- Die Sprache des Besuchers oder der Besucherin (US-Englisch). Auf mehrsprachigen Websites kann dies dazu dienen, die richtige Sprache für einen Besucher auszuwählen.
 - Dass die Anfrage von einer anderen Webseite ausgeht, sowie die URL dieser Seite.
 - Dass es sich bei dem Browser um Chrome auf einem Mac mit dem Betriebssystem OSX handelt. Damit ließe sich festlegen, ob ein Besucher auf eine Desktop- oder Mobilversion der Website weitergeleitet werden soll.

Das Bild zeigt ein Fenster eines Entwicklertools mit der Überschrift "Request Headers". Unterhalb davon sind vier Headerzeilen aufgelistet: "Accept-Language: en-GB,en-US;q=0.9,en;q=0.8", "Referer: http://localhost:8888/phpbook/section_b/intro/index.php", "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36" und eine weitere Zeile, die teilweise abgeschnitten ist. Oben im Fenster befinden sich weitere Registerkarten: "Headers", "Preview", "Response", "Initiator" und "Timing".

Wenn der Webserver eine Anfrage für eine PHP-Seite erhält, **beantwortet** er diese wie folgt:

- Er sucht die in der URL angeforderte PHP-Datei.
- Er veranlasst den PHP-Interpreter, den in der PHP-Datei enthaltenen PHP-Code abzuarbeiten
- Er sendet eine HTML-Seite zurück an den Browser, der die Anfrage gestellt hat

Wenn der Server die HTML-Seite an den Browser zurückschickt, übermittelt er dabei auch **HTTP-Response-Header**; diese enthalten Informationen, die der Browser möglicherweise über die zurückgelieferte Datei wissen muss. Wie die Request-Header verfügt auch jeder Response-Header über einen Namen und einen Wert (ähnlich einer Variablen), die in den Entwicklungswerkzeugen des Browsers angezeigt werden können. Im folgenden Screenshot sendet der Server HTTP-Response-Header, die dem Browser Folgendes mitteilen:

- den Medientyp der Datei und das verwendete Codierungsverfahren (um sicherzustellen, dass die Datei korrekt angezeigt wird)
- Datum und Uhrzeit der Dateiübertragung
- Typ des Webservers, der zum Senden der Datei verwendet wurde

Die Response-Header können aktualisiert werden über:

- die Einstellungen des PHP-Interpreters (siehe die Seiten 196 bis 199)
- die eingebaute Funktion header() (siehe die Seiten 226 bis 227)

Wenn ein Browser die HTML-Datei empfängt, wird sie genauso angezeigt wie jede andere HTML-Seite.

Der Server sendet außerdem noch zwei weitere Datenelemente zurück, die Auskunft darüber geben, ob die Anfrage erfolgreich war oder nicht:

- einen dreistelligen **Statuscode** zur softwaremäßigen Auswertung
- eine für Menschen lesbare **Begründung**

Erfolgreiche Anfragen werden vom Statuscode 200 und der Begründung OK begleitet. Wenn ein Server eine Datei nicht finden kann, lautet der Statuscode 404 und die Begründung Not found. Beim Surfen im Internet haben Sie bestimmt schon einmal so einen Bildschirm gesehen.

Not Found

The requested URL /code/section_b/c5/test.php was not found on this server.

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

In der nachstehenden Tabelle sind die häufigsten Statuscodes und Begründungen aufgeführt. Codes wie 301 (Moved permanently) und 404 (Not Found) helfen Suchmaschinen bei der Indexierung von Websites, wenn sie auf Links zu Seiten stoßen, die gelöscht oder auf eine neue URL verschoben wurden.

STATUSCODE	LESBARE BEGRÜNDUNG
200	OK
301	Moved permanently
307	Temporary redirect
403	Forbidden
404	Not Found
500	Internal server error

x Headers Preview Response Timing

▼ Response Headers view source

TYP & CODIERUNG → Content-Type: text/html; charset=UTF-8
ÜBERTRAGUNGSDATUM → Date: Fri, 15 Jan 2021 15:47:46 GMT
SERVER → Server: Apache/2.4.46 (Unix) OpenSSL/1.0.2u PHP/8.0.0

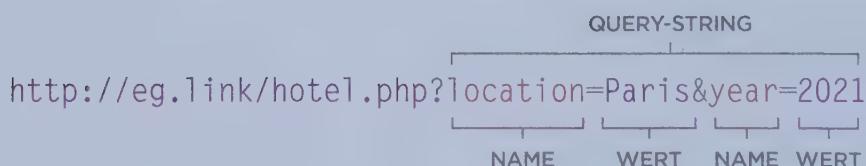
Quelle: Microsoft Corporation, mit freundlicher Genehmigung

WIE DATEN MITTELS HTTP GET UND HTTP POST GESENDET WERDEN

HTTP sieht zwei Möglichkeiten vor, wie Browser Daten an den Server senden können: Bei HTTP GET werden die Daten in den Query-String am Ende der URL eingefügt. Bei HTTP POST werden die Daten in die HTTP-Header eingefügt.

Beim Senden von Daten an eine Webseite mittels **HTTP GET** fügt der Browser die Daten in einen Query-String ein, der am Ende der Seiten-URL angehängt wird. Ein Fragezeichen trennt die Seiten-URL vom Query-String.

Ein Query-String kann mehrere Name/Wert-Paare enthalten. Zwischen Name und Wert steht dabei jeweils ein Gleichheitszeichen. Sollen mehrere Name/Wert-Paare übermittelt werden, dient ein kaufmännisches Und als Trennzeichen.



Beim Senden von Daten über **HTTP POST** fügt der Browser zusätzliche Name/Wert-Paare zu den HTTP-Anfrage-Headern hinzu. Der Browser kann mit jeder Anforderung mehrere Name/Wert-Paare an den Server übermitteln.

Die Header werden nicht im Hauptfenster des Browsers angezeigt, aber in den Entwicklungstools der meisten Browser können Sie diese einsehen. Nachfolgend sehen Sie drei Header und ihre entsprechenden Werte.

Ein Screenshot der Entwicklertools eines Browsers, spezifisch Microsoft Edge DevTools. Die Registerkarte 'Form Data' ist ausgewählt. Es sind drei Formulardatenfelder zu sehen: 'email: ivy@eg.link', 'age: 24' und 'terms: true'. Darunter befindet sich ein Link 'view source' und ein Link 'view URL encoded'.

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

WIE FORMULAR- UND LINK-DATEN VERSENDEN WERDEN

HTML verwendet Links und Formulare, um beim Aufruf einer Seite zusätzliche Daten an den Server zu senden.

Ein Link kann mit einem Query-String zusätzliche Daten an den Server übertragen. Die im Query-String enthaltenen Daten weisen den Server normalerweise an, bestimmte Informationen abzurufen und diese auf der zurückgelieferten Seite anzuzeigen.

Formulare bieten Eingabemöglichkeiten für Text oder Zahlen, die Auswahl einer Listenoption oder das Ankreuzen eines Kästchens. Die Formulardaten können an den Query-String angefügt oder in den HTTP-Header übertragen werden.

OUR HOTELS

[Paris, France](#)

[Oslo, Norway](#)

[Stockholm, Sweden](#)

<http://eg.link/hotel.pnp?location=Oslo>

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

In der Regel kommt HTTP GET zum Einsatz, wenn der Browser Informationen vom Server abrufen möchte, die für alle Website-Besucher gleich sind. Zum Beispiel, wenn sie:

- Links anklicken, um bestimmte Informationen anzuzeigen
- einen Suchbegriff in ein Formular eingeben

Programmierer bezeichnen diese Art von Anfrage manchmal als »sichere Interaktion« (**safe interaction**), weil die Nutzer für die von ihnen durchgeführte Aktion nicht zur Rechenschaft gezogen werden (weil sie z.B. nicht den allgemeinen Geschäftsbedingungen zustimmen oder ein Produkt kaufen).

Username: Ivy

Email: ivy@eg.link

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

In der Regel kommt HTTP POST zum Einsatz, wenn eine Benutzerin Informationen an den Server sendet (oder »postet«), die zu ihrer Identifikation oder zur Aktualisierung der über sie auf dem Server gespeicherten Daten dienen. Zum Beispiel, wenn sie:

- sich in ihrem persönlichen Konto anmeldet
 - ein Produkt kauft
 - einen Dienst abonniert
 - den allgemeinen Geschäftsbedingungen zustimmt
- In diesen Fällen kann die Nutzerin für ihre Handlungen verantwortlich gemacht werden, da sie das Formular ausfüllen und dann bewusst übermitteln muss.

DATENÜBERTRAGUNG VOM ODER ZUM SERVER ABSICHERN

Beim Austausch sensibler Daten zwischen Browser und Server sollten diese **verschlüsselt** werden. Dabei werden die Daten so codiert, dass sie nicht mehr ohne Weiteres mitgelesen werden können. Erst die **Entschlüsselung** bringt die Daten wieder in ein lesbares Format.

Über das Internet verschickte Daten können verschiedene Netze, Router und Server durchqueren, bevor sie ihr Ziel erreichen. Während dieser Reise könnten Unbefugte Zugriff auf die Daten erlangen und versuchen, sie zu mitlesen.

Jede Website, die Informationen über ihre Benutzer sammelt oder deren persönliche Daten auf einer Seite anzeigt, trägt für die sichere Übertragung der Daten zwischen Browser und Server Verantwortung.

Zur sicheren Datenübertragung zwischen Browser und Server nutzen Websites das **HyperText Transfer Protocol Secure (HTTPS)**. HTTPS erweitert HTTP um zusätzliche Regeln für die sichere Datenübertragung zwischen Browsern und Servern.

Um Daten sicher über das Internet zu übertragen, bedarf es der Verschlüsselung. Dabei werden die Daten so verändert, dass sie nicht gelesen werden können, selbst wenn sie auf ihrem Weg abgefangen werden sollten.

Nachrichten werden verschlüsselt, indem die ursprünglichen Zeichen durch eine andere Zeichenfolge ersetzt werden. Dies geschieht unter Verwendung eines Regelwerks, das als **Chiffre** bezeichnet wird.

Der Empfänger muss die Nachricht dann entschlüsseln, um sie wieder lesbar zu machen. Zur Entschlüsselung der Nachricht muss der Empfänger Kenntnis von der Verschlüsselungsmethode haben.

Die zur Entschlüsselung der Nachricht benötigten Informationen werden als **Schlüssel** bezeichnet, da sie die Nachricht »entschlüsseln«.

1. Der Browser verschlüsselt die Formulardaten beim Versenden.

Username:

Email: ivy@ee.link

2. Auf dem Übertragungsweg können die verschlüsselten Daten nicht mitgelesen werden.

NKFAyGCNYKdbNCDTA+XiwR698oP
pAdN1ghyUmRPtkE8y2evzf8LEMe
r0Q89N6XJN2AFt919bAr+qk/qSv
C6b/dRAbb6NqIYXqc6s0IZta/
VZ1UwJTUJH0Io6Qj68+paMgZX/
6wXX0f2VWLxxBM7XwU7ufVZ53
VLQA+mz/wA4jbAFevz8y2f8dbN
CBW2wA

3. Der Server entschlüsselt die Daten mithilfe eines Schlüssels.

Username => Ivy
Email => ivy@
ee.link

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

Um die zwischen Browsern und Servern mittels HTTPS versendeten Daten ver- und entschlüsseln zu können, muss auf dem Webserver ein **Zertifikat** installiert werden. Dieses teilt dem Browser mit, wie er die an den Server übermittelten Informationen verschlüsseln soll.

Um ein Zertifikat zur Installation auf einem Webserver zu erhalten, müssen Sie drei Schritte befolgen:

1. Erstellen Sie eine Zertifikatssignierungsanforderung (**certificate signing request**, kurz: CSR). Dies geschieht auf dem Webserver, der die Website beherbergt. Sie sieht wie eine Reihe von zufälligen Zeichen aus.
2. Kaufen Sie ein Zertifikat von einem Unternehmen, das als autorisierte Zertifizierungsstelle (**certificate authority**, kurz: CA) tätig ist. Diese verlangt von Ihnen ein certificate signing request und Informationen über die Website und ihren Betreiber. Die Zertifizierungsstellen berechnen für das Zertifikat eine Jahresgebühr. Eine Liste der gängigen Zertifizierungsstellen finden Sie hier: <http://notes.re/certificateAuthorities/>.
3. Installieren Sie das Zertifikat auf dem Server, der die Website beherbergt (es handelt sich dabei um eine Textdatei).

HINWEIS: Zertifikate werden nicht immer unverzüglich ausgestellt und müssen daher rechtzeitig vor Inbetriebnahme einer Website beantragt werden.

In der Vergangenheit setzte HTTPS zwei verschiedene Protokolle (Regelwerke) ein, um die über HTTP gesendeten Anfragen und Antworten zu verschlüsseln:

- **Secure Sockets Layer (SSL)**
- **Transport Layer Security (TLS)**

Wenn Sie eine Website lokal mit MAMP oder XAMPP entwickeln, können Sie den Webserver so konfigurieren, dass er auch ohne den Erwerb eines Zertifikats mit HTTPS läuft. Eine Anleitung dazu finden Sie hier: <http://notes.re/local-certificates/>.

Um ein CSR zu erhalten und ein Zertifikat auf den Servern eines Webhosting-Unternehmens zu installieren, sehen Sie in dessen Support-Unterlagen nach.

Sobald ein Zertifikat auf dem Webserver installiert ist und ein Browser eine URL für die Website mit https:// statt mit http:// anfordert, geschieht Folgendes:

- Der Browser verschlüsselt die Anfrage und die HTTP-Request-Header.
- Der Server verschlüsselt die von ihm zurückgelieferte Seite und die HTTP-Response-Header.

Beim Einsatz von https:// zeigen die Browser in der Regel ein Vorhängeschlosssymbol in der Adressleiste an.

Die Begriffe, SSL und TLS, werden oft synonym verwendet, unterscheiden sich jedoch aus technischer Sicht voneinander.

Sie können sich TLS als eine neuere Version von SSL vorstellen. Auf Ihrer Website sollten Sie TLS als Protokoll verwenden.

CODIERUNGSVERFAHREN

Computer verarbeiten Text-, Bild- und Audiodaten in Form von **Binär-informationen**, die aus einer Reihe von Nullen und Einsen bestehen. **Codierungsverfahren** übersetzen die Dinge, die wir Menschen sehen und hören können, in jene computerlesbaren Nullen und Einsen.

Codierungsverfahren spielen eine wichtige Rolle, denn wenn ein Computer das falsche Codierungsverfahren zur Übersetzung zwischen den binären Daten und dem Text, den Bildern und den Audiodaten, die Sie sehen und hören, verwendet, werden die Daten nicht korrekt dargestellt beziehungsweise wiedergegeben.

Der Computer verarbeitet und speichert sämtliche Daten in Form von **Bits (Binärziffern)**. Ein Bit ist eine 0 oder eine 1. Alles in Ihrem Computer (die Buchstaben, die Sie eingeben, die Bilder, die Sie sehen, und die Töne, die Sie hören) wird also in Form von Nullen und Einsen dargestellt. Unten sehen Sie die binäre Entsprechung der einzelnen Buchstaben im Wort HELLO:



01001000 01001010 00101100 00101100 01001111

Wie Sie sehen, werden selbst für einfache Daten eine Menge Bits benötigt. Eine Abfolge von 8 Bits ergibt ein **Byte**.

Codierungsverfahren sind Regeln, nach denen ein Computer Text, Bilder und Audiodaten in Binärdaten (Kombinationen aus Nullen und Einsen) umwandelt, die der Computer verarbeiten und speichern kann.

- Wenn Sie Text eingeben, Bilder hochladen oder Audio aufnehmen, wird mit einem Codierungsverfahren dieser Inhalt in Nullen und Einsen umgewandelt.
- Wenn der Computer Ihnen Text und Bilder anzeigt oder eine Audiodatei abspielt, verwendet er ein Codierungsverfahren, um die Nullen und Einsen in Inhalte zu übersetzen, die Sie sehen oder hören können.

Bildcodierungsverfahren definieren, wie Bilder anhand von Bits dargestellt werden können. Computerbilder bestehen aus einzelnen Quadranten, den sogenannten Pixeln. Unten sehen Sie die Darstellung eines einfachen schwarz-weißen Herzsymbols mit Nullen und Einsen. Jedes weiße Quadrat entspricht einer 0 und jedes schwarze Quadrat entspricht einer 1.

0	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0

Zur Darstellung von Farbbildern muss der Computer wissen, welche Farbe jedes einzelne Pixel hat. Dies erfordert mehr Daten. Verschiedene Bildformate (wie GIF, JPEG, PNG und WebP) verwenden unterschiedliche Codierungsverfahren, um die Farbe jedes Pixels mit Nullen und Einsen darzustellen.

Um das Bild zu verändern, kann ein Computer die Daten der einzelnen Pixel abändern. Zum Beispiel kann ein Filter dazu dienen, jedes Bildpixel abzudunkeln oder aufzuhellen, oder ein Bild kann beschnitten werden, indem Pixel an den Rändern entfernt werden.

Zeichencodierungsverfahren definieren, wie Text mithilfe von Bits dargestellt werden kann, und die einzelnen Zeichencodierungsverfahren können sich auch in der Anzahl der unterstützten Zeichen voneinander unterscheiden. Je mehr Zeichen ein Codierungsverfahren unterstützt, desto mehr Bytes an Daten benötigt es, um diese Zeichen zu verarbeiten.

Wenn Sie eine Website für den internationalen Gebrauch erstellen, sollten Sie ein Codierungsverfahren verwenden, das die Zeichen der Sprachen Ihrer Website-Besucher unterstützt.

ASCII

ASCII ist ein relativ altes Codierungsverfahren, bei dem jedes Zeichen mit 7 Datenbits dargestellt wird. Ein Nachteil von ASCII ist, dass es mit 7 Binärstellen nur 128 mögliche Kombinationen von Nullen und Einsen zulässt, also nicht genug, um alle Zeichen aus allen Sprachen zu unterstützen. Tatsächlich unterstützt ASCII sogar nur 95 Textzeichen.

ISO 8859-1

ISO 8859-1 verwendet 8 Datenbits (1 Byte) zur Darstellung eines Zeichens. Dank des zusätzlichen Datenbits gibt es genügend Kombinationen aus Nullen und Einsen, um sämtliche ASCII-Zeichen sowie die in west-europäischen Sprachen verwendeten Sonderzeichen darzustellen. Es unterstützt jedoch keine Sprachen, die andere Zeichensätze verwenden, wie etwa Chinesisch, Japanisch oder Russisch.

UTF-8

UTF-8 stellt sämtliche Zeichen aller Sprachen dar und ist daher das beste Codierungsverfahren zur Erstellung von Websites.

Zur Unterstützung dieser Sprachen benötigt UTF-8 bis zu vier Bytes an Daten zur Darstellung jedes Zeichens (vier Sätze aus achtstelligen Kombinationen von Nullen und Einsen). Zeichen, die zu ihrer Darstellung mehr als ein Datenbyte benötigen, werden als **Multi-Byte Characters** bezeichnet. Unten sehen Sie zum Beispiel die binären Entsprechungen dreier verschiedener Währungssymbole:

ZEICHEN	BINÄRFOLGE	BYTES
\$	00100100	1
£	11000010 10100011	2
€	11100010 10000010 10101100	3

Es ist wichtig, die Funktionsweise von Codierungsverfahren zu verstehen, da diese:

- an unterschiedlichen Stellen festgelegt werden müssen
- Einfluss darauf haben, welche Zeichen Sie nutzen können
- bestimmen, welche integrierten Funktionen Sie nutzen können

EINGEBAUTE FUNKTIONEN

Einige der im PHP-Interpreter integrierten Funktionen haben einen Parameter zur Angabe des eingesetzten Codierungsverfahrens.

Es gibt auch einige eingebaute Funktionen, die mit einem bestimmten Zeichencodierungsverfahren arbeiten. PHP bietet zum Beispiel eine Funktion zum Ermitteln der Zeichenanzahl in einem String. Diese Funktion wurde in PHP aufgenommen, als ISO 8859-1 das standardmäßige Zeichencodierungsverfahren war. Die Funktion zählte die Anzahl von Bytes, die eine Zeichenkette verwendete (da jedes Zeichen 1 Byte an Daten verbraucht). Als die Unterstützung von UTF-8 in PHP eingeführt wurde, lieferte die Funktion ungenaue Ergebnisse, da einzelne Zeichen nun auch mehr als ein Byte belegen konnten. Daher wurde eine neue Funktion zur Zählung von Multi-Byte-Zeichen integriert.

PHP-EINSTELLUNGEN

Beim Erstellen einer an den Browser zurückzusendenden Seite teilt der PHP-Interpreter dem Browser auch das verwendete Zeichencodierungsverfahren mit, damit dieser die Daten korrekt anzeigen kann. Der PHP-Interpreter verfügt über Einstellungen zur Festlegung dieses Codierungsverfahrens. Ist das Codierungsverfahren nicht korrekt eingestellt, zeigt der Browser für Zeichen, die er nicht erkennt, möglicherweise ein Platzhaltersymbol in Form einer Raute mit einem Fragezeichen ♦ an (oder er stellt sie überhaupt nicht dar).

CODE-EDITOREN

Da die PHP-Dateien selbst aus Text bestehen, können Sie in den meisten Code-Editoren das Zeichencodierungsverfahren angeben, das zum Speichern der PHP-Dateien verwendet werden soll.

Der nachfolgende Link zeigt Ihnen, wie Sie das Zeichencodierungsverfahren in einigen beliebten Code-Editoren einstellen können:
<http://notes.re/editors/set-encoding>.

Wenn Sie irgendwo die Option »UTF-8 ohne BOM« sehen, sollten Sie diese auswählen.

Im Buchteil C erfahren Sie zudem, dass Datenbanken das Codierungsverfahren einer Website kennen müssen.

DAS IM PHP-INTERPRETER INTEGRIERTE TOOLKIT

In diesem Buchteil lernen Sie immer wieder in den PHP-Interpreter integrierte Werkzeuge kennen, die Ihnen bei der Erstellung dynamischer Webseiten helfen.

Sie wissen bereits, dass der PHP-Interpreter ein Programm ist, das auf einem Webserver läuft.

Wenn Sie ein Programm auf einem Desktop- oder Laptop-Computer öffnen (z. B. eine Textverarbeitungs- oder Bildbearbeitungssoftware), gibt es eine grafische Benutzeroberfläche (GUI). Die Symbolleisten und Menüoptionen der grafischen Benutzeroberfläche dienen zur Ausführung der Aufgaben, für die die Software entwickelt wurde, und die Ergebnisse werden Ihnen dann auf dem Bildschirm angezeigt.

Der PHP-Interpreter hat keine grafische Benutzeroberfläche. Stattdessen verfügt er über eine Reihe von eingebauten Arrays, Funktionen und Klassen, mit deren Hilfe der PHP-Code in Ihren PHP-Dateien jene Aufgaben ausführen kann, die bei der Verarbeitung von Daten und der Erstellung von HTML-Seiten zur nachfolgenden Übermittlung an den Browser typischerweise anfallen.

Der PHP-Interpreter greift außerdem auf Textdateien zurück, um Optionen und Einstellungen zu steuern und mögliche Fehler zu protokollieren.

SUPERGLOBALE ARRAYS

Seiten 190 bis 191

Immer wenn ein Browser eine PHP-Seite aufruft, erstellt der PHP-Interpreter eine Reihe so- genannter **superglobaler Arrays**. Diese enthalten Daten, auf die der PHP-Code dieser Seite zugreifen und die er verwenden kann.

Alle superglobalen Arrays sind assoziative Arrays, daher müssen Sie wissen:

- wie die Arrays heißen
- welche Schlüssel die einzelnen Arrays verwenden
- welche Daten jeder dieser Schlüssel speichert

Sobald der PHP-Interpreter die HTML-Seite erstellt und an den Browser zurückgesendet hat, werden die Daten in diesen Arrays automatisch verworfen, da sie sich nur auf die jeweilige Seitenanforderung beziehen.

Wenn die Datei das nächste Mal ausgeführt wird, kann sie wieder auf eine Reihe superglobaler Arrays mit Daten zugreifen, die sich nur auf diese spezielle Anfrage beziehen.

EINGEBAUTE FUNKTIONEN

Seiten 192 bis 193 und Kapitel 5

Die **eingebauten Funktionen** können Sie mit den Menübefehlen einer grafischen Benutzeroberfläche vergleichen. Eine Funktion durchsucht beispielsweise eine Zeichenkette nach einer bestimmten Zeichenfolge und ersetzt sie durch eine andere, ähnlich wie die Funktion »Suchen und Ersetzen« in einem Textverarbeitungsprogramm. Statt einen Menüeintrag in der grafischen Benutzeroberfläche zu nutzen, rufen Sie die Funktion in Ihrem PHP-Code auf.

In Kapitel 3 haben Sie gelernt, eine Funktion zu definieren und auch aufzurufen. Eingebaute Funktionen rufen Sie auf die gleiche Weise auf, aber Sie brauchen keine Funktionsdefinition in die Seite einzufügen, da sie bereits in den PHP-Interpreter eingebaut ist.

Um die eingebauten Funktionen verwenden zu können, müssen Sie Folgendes wissen:

- den Namen der Funktion
- den/die erforderlichen Parameter
- den Rückgabewert der Funktion (oder was sie auf der Seite anzeigt)

EINGEBAUTE KLASSEN

Seiten 318 bis 327

Mit den **eingebauten Klassen** lassen sich Objekte erstellen, die in der Programmierung für häufig verwendete Dinge stehen. Die Klasse `DateTime` dient beispielsweise zur Erstellung von Objekten, die Datums- und Zeitangaben darstellen. Die Klasse verfügt über Eigenschaften und Methoden, um mit den Datums- und Zeitangaben, die ein mit dieser Klasse erstelltes Objekt darstellt, umzugehen.

In Kapitel 4 haben Sie erfahren, wie Sie Klassendefinitionen schreiben und zur Erstellung von Objekten verwenden. Wenn Sie ein Objekt mit einer eingebauten Klasse erstellen, brauchen Sie keine Klassendefinition in die Seite einzufügen, da diese ja bereits in den PHP-Interpreter integriert ist.

Um die eingebauten Klassen zu verwenden, müssen Sie wissen, wie Sie ein Objekt mit dieser Klasse erstellen und:

- welche Eigenschaften sie hat
- welche Methoden sie hat
- welche Parameter jede Methode hat
- welchen Wert jede Methode zurückgibt

FEHLERmeldungen

Seiten 194 bis 195 und Kapitel 10

Wenn der PHP-Interpreter im Code auf Fehler stößt, erzeugt er eine Fehlermeldung.

Bei der Entwicklung einer Website sollten Fehlermeldungen in der vom PHP-Interpreter an den Browser zurückgesendeten Seite angezeigt werden. Auf diese Weise können die Entwickler mögliche Fehler beim Ausführen der Seite direkt erkennen.

Sobald eine Website live geht, sollten die Fehler für die Besucher nicht mehr sichtbar sein. Da die Entwickler die Fehler nicht in Echtzeit sehen können (sie können nicht allen Benutzern über die Schulter schauen), werden die Fehlermeldungen in Textform in einer sogenannten **Protokolldatei** gespeichert und auf dem Server abgelegt. Die Entwickler gehen dann diese **Log Files** durch, um festzustellen, ob Fehler aufgetreten sind, die während der Entwicklung der Website übersehen wurden.

EINSTELLUNGEN

Seiten 196 bis 199

Desktop-Programme haben oft Menüoptionen, die zu Fenstern mit Einstellungen oder Optionen für die Funktionsweise der Software führen. In den Einstellungen eines Textverarbeitungsprogramms lässt sich beispielsweise das Papierformat oder die Standardsprache eines Dokuments auswählen.

Da der PHP-Interpreter und der Webserver keine grafische Benutzeroberfläche haben, werden ihre Einstellungen in speziellen Textdateien festgelegt. Zum Beispiel gibt es Einstellungen, die steuern, ob der PHP-Interpreter die Fehlermeldungen auf dem Bildschirm anzeigen oder in einer Logdatei speichern soll und wo die Fehlerprotokolldatei auf dem Server liegen soll.

Diese Textdateien können Sie mit demselben Code-Editor bearbeiten, mit dem Sie die PHP-Seiten erstellen.

SUPERGLOBALE ARRAYS

Das superglobale Array `$_SERVER` ist ein Beispiel für die superglobalen Arrays, die der PHP-Interpreter bei jeder Seitenanforderung neu erstellt. Jedes superglobale Array enthält Daten, auf die der PHP-Code der Seite zugreifen kann.

Bei allen superglobalen Arrays handelt es sich um assoziative Arrays. Sie müssen den Namen des Arrays, die Schlüssel jedes einzelnen Arrays und die darin enthaltenen Daten kennen.

Unten sehen Sie, was das superglobale Array `$_SERVER` beinhaltet. Es speichert Informationen:

- über den Browser (die in den HTTP-Headern übermittelt werden)
- über den Typ der HTTP-Anfrage (GET oder POST)
- über die angeforderte URL
- über den Speicherort der Datei auf dem Server

Die im superglobalen Array gespeicherten Daten werden genauso abgerufen wie beim Zugriff auf ein assoziatives Array. Um die IP-Adresse des Browsers in der Variablen `$ip` zu speichern, nutzen Sie folgenden Ausdruck:

SUPERGLOBALES ARRAY
 |
\$ip = \$_SERVER['REMOTE_ADDR'];
 |
SCHLÜSSEL

SCHLÜSSEL	ZWECK
<code>\$_SERVER['REMOTE_ADDR']</code>	IP-Adresse des Browsers
<code>\$_SERVER['HTTP_USER_AGENT']</code>	Browsertyp, mit dem die Seite aufgerufen wurde
<code>\$_SERVER['HTTP_REFERER']</code>	Wenn der Besucher über einen Link auf diese Seite gelangt ist, kann ein Browser die URL der Seite übermitteln, die auf diese Seite verlinkt hat. Nicht alle Browser senden diese Daten.
<code>\$_SERVER['REQUEST_METHOD']</code>	Art der HTTP-Anfrage: GET oder POST
<code>\$_SERVER['HTTPS']</code>	Wird dem Array nur hinzugefügt, wenn der Seitenaufruf über HTTPS erfolgt; in diesem Fall ist der Wert <code>true</code> .
<code>\$_SERVER['HTTP_HOST']</code>	Der Hostname (ein Domänenname, eine IP-Adresse oder <code>localhost</code>)
<code>\$_SERVER['REQUEST_URI']</code>	Die URI (nach dem Hostnamen), die zum Anfordern dieser Seite verwendet wird
<code>\$_SERVER['QUERY_STRING']</code>	Eventuelle Daten im Query-String
<code>\$_SERVER['SCRIPT_NAME']</code>	Pfad vom Stammverzeichnis des Dokuments zur gerade ausgeführten Datei
<code>\$_SERVER['SCRIPT_FILENAME']</code>	Pfad vom Stammverzeichnis des Dateisystems zur gerade ausgeführten Datei
<code>\$_SERVER['DOCUMENT_ROOT']</code>	Pfad vom Stammverzeichnis des Dateisystems zum Stammverzeichnis des Dokuments der gerade ausgeführten Datei

Der Unterschied zwischen dem Stammverzeichnis eines Dokuments und dem Stammverzeichnis eines Dateisystems wird hier beschrieben: <http://notes.re/php/filepaths>

DATEN IM SUPERGLOBALEN ARRAY \$_SERVER

Jedes Element des superglobalen Arrays `$_SERVER` enthält eine andere Information über den Seitenaufruf oder die angeforderte Datei.

Im Folgenden werden die einzelnen im superglobalen Array `$_SERVER` gespeicherten Werte durch Auswahl der Daten in den verschiedenen Array-Schlüsseln ausgegeben.

PHP

section_b/intro/server-superglobal.php

```
<table>
  <tr><th colspan="2" class="title">In HTTP-Headern gesendete Browserdaten </th></tr>
  <tr><th>IP-Adresse des Browsers </th><td><?= $_SERVER['REMOTE_ADDR'] ?> </td></tr>
  <tr><th>Browsertyp </th><td><?= $_SERVER['HTTP_USER_AGENT'] ?></td></tr>
  <tr><th colspan="2" class="title">HTTP Request </th></tr>
  <tr><th>Hostname </th><td><?= $_SERVER['HTTP_HOST'] ?> </td></tr>
  <tr><th>URI nach Hostname </th><td><?= $_SERVER['REQUEST_URI'] ?> </td></tr>
  <tr><th>Query-String </th><td><?= $_SERVER['QUERY_STRING'] ?> </td></tr>
  <tr><th>HTTP-Request-Methode </th><td><?= $_SERVER['REQUEST_METHOD'] ?> </td></tr>
  <tr><th colspan="2" class="title">Speicherort der ausgeführten Datei </th></tr>
  <tr><th>Stammverzeichnis Dokument </th><td><?= $_SERVER['DOCUMENT_ROOT'] ?> </td></tr>
  <tr><th>Pfad vom Stammverzeichnis Dokument </th><td><?= $_SERVER['SCRIPT_NAME'] ?> </td></tr>
  <tr><th>Absoluter Pfad </th><td><?= $_SERVER['SCRIPT_FILENAME'] ?></td></tr>
</table>
```

ERGEBNIS

DATA ABOUT BROWSER SENT IN HTTP HEADERS	
BROWSER'S IP ADDRESS	127.0.0.1
TYPE OF BROWSER	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.115 (KHTML, like Gecko) Version/14.0.2 Safari/605.115
HTTP REQUEST	
HOST NAME	localhost:8888
UNCHANGED HOST NAME	/phpbook/section_b/intro/server-superglobal.php
QUERY STRING	
UNCHANGED REQUEST METHOD	GET
LOCATION OF THE FILE BEING EXECUTED	
DOCUMENT ROOT	/Users/jon/Sites/localhost
PATH FROM DOCUMENT ROOT	/phpbook/section_b/intro/server-superglobal.php
ABSOLUTE PATH	/Users/jon/Sites/localhost/phpbook/section_b/intro/server-superglobal.php

EINGEBAUTE FUNKTIONEN ZUR ANZEIGE VON DATEN IN VARIABLEN

Die Funktion `var_dump()` ist ein Beispiel für eine der eingebauten PHP-Funktionen. Sie wird bei der Website-Entwicklung eingesetzt, um zu überprüfen, welche(n) Wert(e) eine Variable enthält und welchen Datentyp sie aufweist.

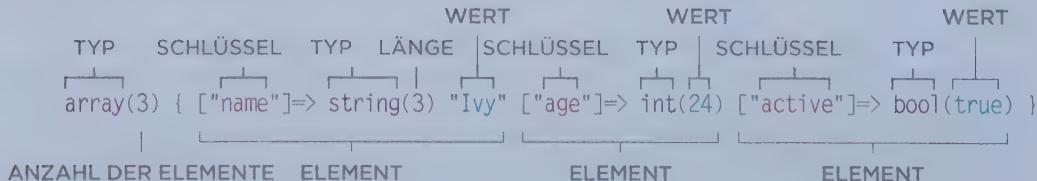
Um eine integrierte Funktion zu verwenden (oder aufzurufen), müssen Sie nur ihren Namen, ihre Parameter und die Werte kennen, die sie zurückgibt beziehungsweise auf der Seite anzeigt.

`var_dump()` hat nur einen Parameter: den Variablennamen.

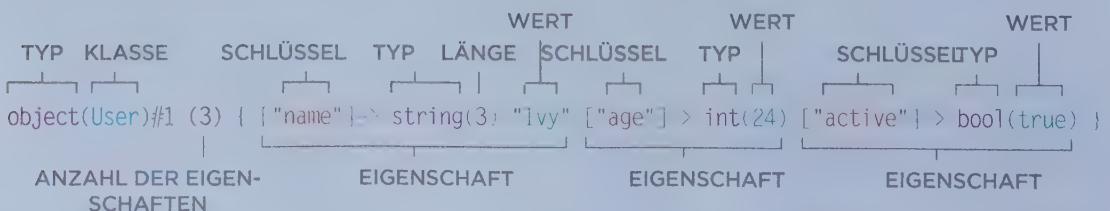
Die Funktion gibt keinen Wert zurück; sie zeigt die in der Variablen gespeicherten Werte auf der erstellten HTML-Seite an.

```
var_dump($variable);
```

Enthält die Variable ein Array, zeigt die Funktion das Wort `array` gefolgt von der Anzahl der Elemente in Klammern an.

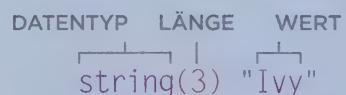


Wenn die Variable ein Objekt enthält, werden das Wort `object`, der Klassenname und die Anzahl der Eigenschaften angezeigt.

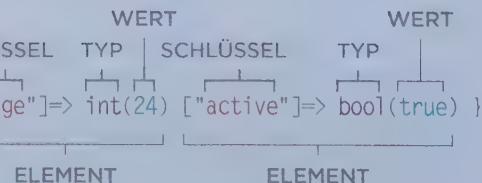


Wenn die Variable einen Skalarwert (eine Zeichenkette, Zahl oder Ganzzahl) enthält, zeigt sie den Datentyp und den Wert an

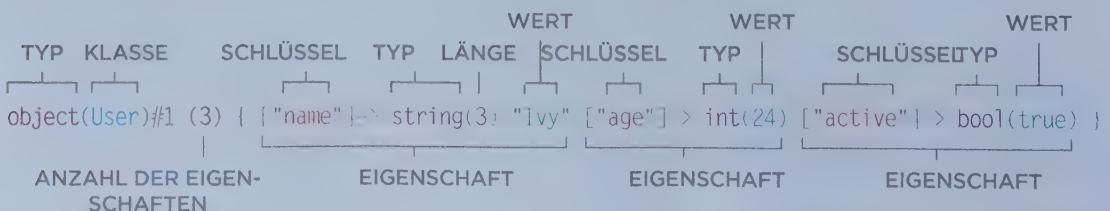
Handelt es sich bei dem Wert um einen String, wird die Anzahl der Zeichen in der Zeichenkette hinter dem Datentyp in Klammern angezeigt.



Es folgen dann in geschweiften Klammern der Schlüssel, der Datentyp des Werts und der Wert eines jedes Elements.



Für jede Eigenschaft werden der Name, der Datentyp des Werts und der Wert angezeigt. (Es werden keine Methoden angezeigt.)



DIE INHALTE EINER VARIABLEN ANZEIGEN

PHP

section_b/intro/var-dump.php

```
<?php  
① $username = 'Ivy';  
  
② $user_array = [  
    'name' => 'Ivy',  
    'age'  => 24,  
    'active' => true,  
];  
  
③ class User  
{  
    public $name;  
    public $age;  
    public $active;  
    public function __construct($name, $age, $active) {  
        $this->name = $name;  
        $this->age = $age;  
        $this->active = $active;  
    }  
}  
  
④ $user_object = new User('Ivy', 24, true);  
?>  
  
...  
⑤ <p>Scalar: <?php var_dump($username); ?></p>  
<p>Array:   <?php var_dump($user_array); ?></p>  
<p>Object:  <?php var_dump($user_object); ?></p>
```

ERGEBNIS

```
Scalar: string(3) "Ivy"  
  
Array: array(3) { [0]=> string(3) "Ivy" [1]=> int(24) [2]=> bool(true)  
  
Object: object(User) {#111} {"name":=> string(3) "Ivy", "age":=> int(24), "active":=> bool(true)}
```

In diesem Beispiel werden Variablen zum Speichern eines Skalarwerts, eines Arrays und eines Objekts erstellt. Anschließend wird mit `var_dump()` angezeigt, was die einzelnen Variablen enthalten.

1. Die Variable `$username` speichert den Namen eines Website-Mitglieds als String.
2. Die Variable `$user_array` enthält ein Array zur Speicherung von Name, Alter und Aktivitätsstatus des Mitglieds.
3. Die neu erstellte Klasse `User` dient als Vorlage für Objekte, die die Mitglieder einer Website darstellen. Sie verfügt über drei Eigenschaften: den Benutzernamen, das Alter und den Aktivitätsstatus.
4. Ein Objekt wird mithilfe der Klasse `User` erstellt und in der Variablen `$user_object` gespeichert.

5. Die in den Variablen gespeicherten Werte werden mit der Funktion `var_dump()` angezeigt.

HINWEIS: Wenn Sie jeden PHP-Block mit `<pre>`-HTML-Tags umgeben, werden die angezeigten Daten auf einzelne Zeilen verteilt, was die Lesbarkeit verbessert.

```
<pre>  
<?php var_dump($username) ?>  
</pre>
```

FEHLERQUELLEN

Bei Problemen mit dem PHP-Code erzeugt der PHP-Interpreter entsprechende Fehlermeldungen, die Ihnen die Fehlersuche erleichtern.

Wenn der PHP-Interpreter bei der Code-Ausführung auf ein Problem stößt, erzeugt er eine Fehlermeldung. Es gibt zwei Möglichkeiten, die Meldungen zu betrachten; sie können:

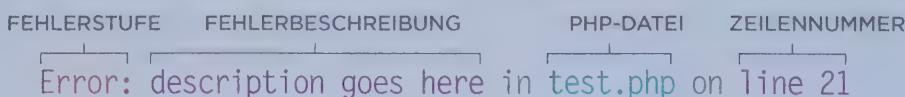
- im an den Browser übermittelten HTML-Code angezeigt werden
- in einer Textdatei, dem sogenannten **Error Log**, gespeichert werden

Jede Fehlermeldung umfasst vier Elemente, die Ihnen helfen, die Fehlerursache zu finden und zu beheben:

- die **Fehlerstufe** (oder den Schweregrad des Fehlers; die Stufen sind in der nachstehenden Tabelle beschrieben)
- eine Beschreibung des Fehlers
- die Datei, in der der Fehler aufgetreten ist
- die Zeilennummer, wo der Fehler gefunden wurde

Während der Entwicklung einer Website sollten Fehlermeldungen in den HTML-Seiten ausgegeben werden, die der PHP-Interpreter an den Browser zurückschickt, damit das Entwicklungsteam sie sofort sehen kann. Wenn eine Website live ist, sollten sie in einem Fehlerprotokoll (einer Textdatei auf dem Server) angelegt werden; die Besucher sollten sie dagegen nicht mehr zu Gesicht bekommen. Wie Sie diese Einstellung ändern können, erfahren Sie auf den Seiten 352 und 353.

Der Code für diesen Buchteil gibt Fehler auf der HTML-Seite aus. Wenn Sie die Übungen, die auf viele der Beispiele in diesem Abschnitt folgen, ausprobieren, lassen Sie sich nicht von möglichen Fehlermeldungen entmutigen. Diese Mitteilungen helfen Ihnen, herauszufinden, wo es Probleme mit Ihrem Code gibt und wie Sie diese beheben können. Mehr über die Fehlerbehandlung erfahren Sie in Kapitel 10.



STUFE	BESCHREIBUNG
PARSE	Syntaxfehler im PHP-Code verhindern, dass der PHP-Interpreter überhaupt versucht, die Seite auszuführen.
FATAL	Ein Fehler im PHP-Code, der die Ausführung von jedem nachfolgenden Code verhindert
WARNING	Dies wird wahrscheinlich ein Problem verursachen, aber der Interpreter versucht, den Rest der Seite dennoch auszuführen.
NOTICE	Dies könnte auf ein Problem hindeuten, aber der Interpreter versucht, den Rest der Seite dennoch auszuführen.
DEPRECATED	PHP-Code, der wahrscheinlich aus zukünftigen PHP-Versionen entfernt werden wird
STRICT	PHP-Code, der besser geschrieben werden könnte und somit zukunftssicherer ist

BEISPIELE FÜR FEHLERMELDUNGEN

Fehlermeldungen wirken auf den ersten Blick oft kryptisch, aber die darin enthaltenen Informationen helfen Ihnen herauszufinden, was mit Ihrem Code nicht stimmt.

PHP

section_b/intro/error1.php

```
<?php  
① echo $name;  
② echo ' welcome to our site.';  
?>
```

ERGEBNIS

Warning: Undefined variable \$name in
/Users/Jon/Sites/localhost/phpbook/section_b/intro/error1.php on line 2
welcome to our site.

PHP

section_b/intro/error2.php

```
<?php  
③ echo 'Hello ';  
username = 'Ivy';  
?>
```

ERGEBNIS

Parse error: syntax error, unexpected token "=" in
/Users/Jon/Sites/localhost/phpbook/section_b/intro/error2.php on line 3

1. Die Seite versucht, eine Variable auszugeben, die noch gar nicht erstellt wurde. Eine Warnung besagt, dass in error1.php in Zeile 2 die nicht definierte Variable \$name auftaucht. Da die Fehlerstufe Warning lautet, läuft der Interpreter weiter.

(Bis PHP 7.4 lieferte dieser Fehler ein Notice und kein Warning.)

2. Die Seite gibt den Text welcome to our site aus (den Sie in der letzten Zeile des Ergebniskastens sehen können).

3. Die echo-Anweisung würde das Wort Hello anzeigen, allerdings erzeugt die nächste Zeile einen Parse-Fehler, der den PHP-Interpreter daran hindert, den Code auf der Seite auszuführen.

Der Fehler wird durch ein fehlendes \$-Symbol am Anfang der Variablen \$username verursacht.

Mehr über die Fehlerbehandlung und Fehlermeldungen erfahren Sie in Kapitel 10.

EINSTELLUNGEN & OPTIONEN FÜR DEN PHP-INTERPRETER

Die Einstellungen und Vorgaben für Software auf einem Desktop-Computer werden häufig über ein Menü in der Benutzeroberfläche vorgenommen. Die Einstellungen des PHP-Interpreters und des Apache-Webservers werden mithilfe von Textdateien festgelegt.

Der Apache-Webserver und der PHP-Interpreter verfügen über Einstellungen, die zum Beispiel die standardmäßige Zeichencodierung, die Anzeige von Fehlermeldungen bei Problemen und den maximal zulässigen Speicherbedarf einer einzelnen Webseite, festlegen. Die Dateien zur Verwaltung dieser Einstellungen können Sie in Ihrem Code-Editor bearbeiten.

php.ini

Die Textdatei `php.ini` enthält die Standardeinstellungen des PHP-Interpreters. Sie können die Einstellungen in dieser Datei verändern, dürfen sie aber nicht entfernen. Nachdem Sie Änderungen vorgenommen haben, müssen Sie den Webserver neu starten, damit diese wirksam werden.

Um herauszufinden, wo sich die Datei `php.ini` befindet, verfügt PHP über die eingebaute Funktion `phpinfo()`. Sie zeigt die Einstellungen des PHP-Interpreters in einer HTML-Tabelle an (siehe rechte Seite). Der Pfad zur Datei `php.ini` steht in der ersten Tabelle neben dem Titel **Loaded configuration file**.

Einige Webhoster verweigern Ihnen den Zugriff auf die Datei `php.ini`, da sie meist steuert, wie sämtliche PHP-Dateien auf dem Webserver ausgeführt werden (und sich daher jegliche Änderung auch auf andere Websites auf demselben Server auswirken würde). Wenn Sie keinen Zugang auf `php.ini` haben, können Sie mittels `.htaccess` viele der gleichen Einstellungen beeinflussen.

Wenn eine Datei andere Einstellungen benötigt als die übrigen PHP-Dateien auf dem Webserver, können Sie die eingebaute PHP-Funktion `ini_set()` verwenden, um einige der Einstellungen in der Datei `php.ini` zu überschreiben.

httpd.conf

Die Textdatei `httpd.conf` definiert die Standardeinstellungen des Apache-Webservers. Einige Einstellungen überschneiden sich mit denen in `php.ini`, und Apache muss neu gestartet werden, um mögliche Änderungen wirksam werden zu lassen. Webhoster gewähren ihren Kunden nicht immer Zugriff auf `httpd.conf`, da diese Datei meist die Einstellungen des gesamten Webservers steuert.

.htaccess

Apache-Nutzer können Dateien mit der Bezeichnung `.htaccess` zu einem beliebigen Ordner im Dokumentstamm einer Website hinzuzufügen. Die Regeln in einer `.htaccess`-Datei gelten dabei nur für die Dateien in demselben Verzeichnis wie die `.htaccess`-Datei sowie für all seine Unterordner und ersetzen die Einstellungen in `httpd.conf` und `php.ini`.

Änderungen an `.htaccess`-Dateien werden sofort nach dem Speichern der Datei wirksam. Sie sollten `.htaccess` jedoch nur verwenden, wenn `httpd.conf` oder `php.ini` nicht infrage kommen, da dieser Weg langsamer ist als die Änderung der Standardeinstellungen. Die meisten Webhosting-Unternehmen gestatten Ihnen die Erstellung von `.htaccess`-Dateien, schränken aber möglicherweise ein, welche Einstellungen Sie darin treffen können (z.B. die maximale Größe der hochgeladenen Dateien).

Betriebssysteme behandeln `.htaccess`-Dateien als versteckte Dateien, sodass Sie Ihren Dateimanager oder Ihr FTP-Programm möglicherweise so einstellen müssen, dass diese dennoch angezeigt werden. Wie Sie sich versteckte Dateien anzeigen lassen, erfahren Sie unter: http://notes.re/hidden_files.

Der Code für dieses Buch verwendet mehrere `.htaccess`-Dateien, sodass für verschiedene Kapitel unterschiedliche Einstellungen gelten können.

EINSTELLUNGEN FÜR DEN PHP-INTERPRETER BETRACHTEN

Wie die meisten Programme verfügt auch der PHP-Interpreter über Einstellungen (oder Vorgaben) zur Steuerung seiner Arbeitsweise. Die eingebaute Funktion `phpinfo()` zeigt Tabellen der zu konfigurierenden Einstellungen und ihrer aktuellen Werte an.

PHP

section_b/intro/phpinfo.php

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

ERGEBNIS



Die Funktion `phpinfo()` erstellt eine lange Reihe von Tabellen, die die Einstellungen für den PHP-Interpreter und ihre Standardwerte enthalten. Diese Einstellungen gelten für jede PHP-Datei, die der PHP-Interpreter ausführt.

Die auf der linken Seite vorgestellten Textdateien können verwendet werden, um diese Einstellungen zu ändern.

Die folgende Tabelle beschreibt einige der Einstellungen, die Sie in diesem Buchteil zu kontrollieren lernen müssen. Sofern nicht anders angegeben, finden Sie diese unter der Überschrift Core.

EINSTELLUNG	BESCHREIBUNG
<code>default_charset</code>	Standard-Zeichencodierung. Sollte auf UTF-8 eingestellt sein.)
<code>display_errors</code>	Fehler auf der HTML-Seite ein-/ausschalten. Ein für die Entwicklung. Aus, wenn die Seite live geht.
<code>log_errors</code>	Fehlerprotokolldatei ein-/ausschalten. Ein, wenn die Website online geht.
<code>error_log</code>	Pfad zur Fehlerprotokolldatei, wenn die Website online geht.
<code>error_reporting</code>	Welche Fehler aufgezeichnet werden sollen. (Die Einstellung <code>E_ALL</code> erfasst alle Fehler.)
<code>upload_max_filesize</code>	Maximale Größe einer Einzeldatei, die ein Browser auf den Server hochladen kann.
<code>max_execution_time</code>	Maximale Anzahl von Sekunden, die ein Skript laufen kann, bevor der PHP-Interpreter es abbricht.
<code>date.timezone</code>	Standardzeitzone, die vom Server verwendet wird. Wird unter der Überschrift Date angezeigt.

INTERPRETER-EINSTELLUNGEN ÄNDERN: PHP.INI

Mit der Datei `php.ini` können Sie die Einstellungen des PHP-Interpreters bearbeiten. Sie dürfen die darin enthaltenen Werte nur bearbeiten und keine der vorhandenen Einstellungen löschen.

Die Datei `php.ini` ist recht lang, da sie alle Einstellungen für den PHP-Interpreter enthält. Dazu kommen noch zahlreiche Kommentare zur Erklärung der Einstellungen. Alles, was hinter einem Semikolon steht, ist ein Kommentar.

Die Einstellungen werden über **Direktiven** vorgenommen, die wie Variablen sind. Bearbeiten Sie nur die Werte der Direktiven (und löschen Sie keine Zeilen). Um eine Einstellung zu finden, die Sie beeinflussen wollen, öffnen Sie die Datei und suchen nach dieser Einstellung.

Eine vollständige Liste der Direktiven finden Sie unter:
<http://php.net/manual/en/ini.list.php>

Jede Direktive beginnt auf einer neuen Zeile und besteht aus:

- einer zu verändernden Option
 - einem Zuweisungsoperator
 - dem gewünschten Wert

Wenn der Wert:

- eine Zeichenkette ist, setzen Sie ihn in Anführungszeichen
 - eine Zahl ist, verzichten Sie auf Anführungszeichen
 - ein boolescher Wert ist, verzichten Sie auf Anführungszeichen

```
date.timezone  = "Europe/Rome"  
display_errors = On
```

php.ini (nicht Bestandteil der Download-Dateien)

PHP

```
; Eine kommentierte Auswahl der veränderbaren Werte in der Datei ini.php
default_charset      = "UTF-8" ; standardmäßig verwendeter Zeichensatz
display_errors       = On   ; ob Fehler auf der Seite angezeigt werden sollen oder nicht
log_errors          = On   ; Fehler in ein Fehlerprotokoll schreiben
error_reporting     = E_ALL ; alle Fehler anzeigen
upload_max_filesize = 32M  ; maximale Größe eines Dateiuploads
post_max_size        = 32M  ; maximale Datenmenge zum Senden über HTTP POST
max_execution_time  = 30   ; maximale Ausführungsduer je Skript in Sekunden
memory_limit         = 128M ; maximaler Speicherbedarf je Skript
date.timezone        = "Europe/Rome" ; standardmäßige Zeitzone
```

INTERPRETER-EINSTELLUNGEN VERÄNDERN: .HTACCESS

.htaccess-Dateien können zu jedem Verzeichnis auf einem Apache-Webserver hinzugefügt werden. Sie überschreiben die Einstellungen des PHP-Interpreters für alle in diesem Ordner oder seinen Unterordnern befindlichen Dateien.

Eine .htaccess-Datei muss nur jene Einstellungen enthalten, die Sie in der php.ini überschreiben wollen. Die Einstellungen haben die gleichen Namen und Werte wie die in php.ini, außer dass ihnen Folgendes vorangestellt wird:

- `php_flag`, wenn die Direktive einen booleschen Wert enthält (also eine Einstellung, die den Wert `on` oder `off` annehmen kann)
- `php_value`, wenn es mehr als zwei mögliche Werte gibt (z.B. für Zahlen, Orte oder Codierungsverfahren)

Kommentare beginnen mit einem # -Symbol und müssen auf einer neuen Zeile stehen (nicht in derselben Zeile wie die Direktive).

```
php_value date.timezone "Europe/London"
php_flag display_errors On
```

TYP AUF OPTION WERT

PHP

section_b/intro/.htaccess

```
# Beispiel-.htaccess zur Verwendung mit Beispielen (Optionen im php.ini-Beispiel beschrieben)
php_value default_charset "UTF-8"
php_flag display_errors On
php_flag log_errors Off
php_value error_reporting -1
php_value upload_max_filesize 32M
php_value post_max_size 32M
php_value max_execution_time 30
php_value memory_limit 128M
php_value date.timezone "Europe/London"
```

KAPITEL IN TEIL B

DYNAMISCHE WEBSEITEN

5

EINGEBAUTE FUNKTIONEN

Jede der eingebauten PHP-Funktionen erfüllt eine bestimmte Aufgabe, die Programmierer beim Umgang mit Daten häufig benötigen. Die eingebauten Funktionen werden am Anfang vorgestellt, weil sie in allen weiteren Buchkapiteln Verwendung finden.

6

DATEN VON BROWSERN ERHALTEN

Dieses Kapitel zeigt, wie der PHP-Interpreter auf die vom Browser gesendeten Daten zugreifen kann, wie er prüft, ob die von der Seite benötigten Daten zur Verfügung gestellt wurden und ob sie im richtigen Format vorliegen. Sie lernen auch, wie Sie die vom Benutzer bereitgestellten Daten sicher auf einer Seite anzeigen können.

7

BILDER & DATEIEN

Wenn Sie Nutzern die Möglichkeit geben, Bilder oder andere Dateien an eine Website zu senden, müssen Sie wissen, wie der PHP-Interpreter diese Dateien behandelt. In diesem Kapitel erfahren Sie auch, wie Sie die Größe von Bildern ändern und kleinere Miniaturbildversionen erstellen können.

8

DATUM & UHRZEIT

Datums- und Zeitangaben werden auf viele unterschiedliche Weisen geschrieben, sodass Sie wissen müssen, wie Sie diese mittels PHP einheitlich formatieren können. Sie lernen auch, wie Sie typische Anforderungen wie die Darstellung von Zeitintervallen oder die Behandlung wiederkehrender Ereignisse umsetzen können.

9

COOKIES & SITZUNGEN

Dieses Kapitel zeigt, wie Textdateien, sogenannte Cookies, im Browser hinterlegt werden können, um Informationen über den jeweiligen Besucher zu speichern. Sie erfahren auch, wie Sie mithilfe von Sessions Informationen für kurze Zeit auf dem Webserver speichern (z.B. während eines einzelnen Besuchs auf einer Website).

10

FEHLERBEHANDLUNG

Jeder macht beim Programmieren auch mal Fehler, die zu Fehlermeldungen des PHP-Interpreters führen. In diesem Kapitel erfahren Sie, wie Sie diese Fehlermeldungen richtig deuten und mit welchen Arbeitstechniken Sie Fehler in Ihrem Code finden und beheben.

5

EINGEBAUTE FUNKTIONEN

Dieses Kapitel beschreibt eine Reihe von Funktionen, die fester Bestandteil des PHP-Interpreters sind. Jede Funktion übernimmt eine bestimmte Aufgabe.

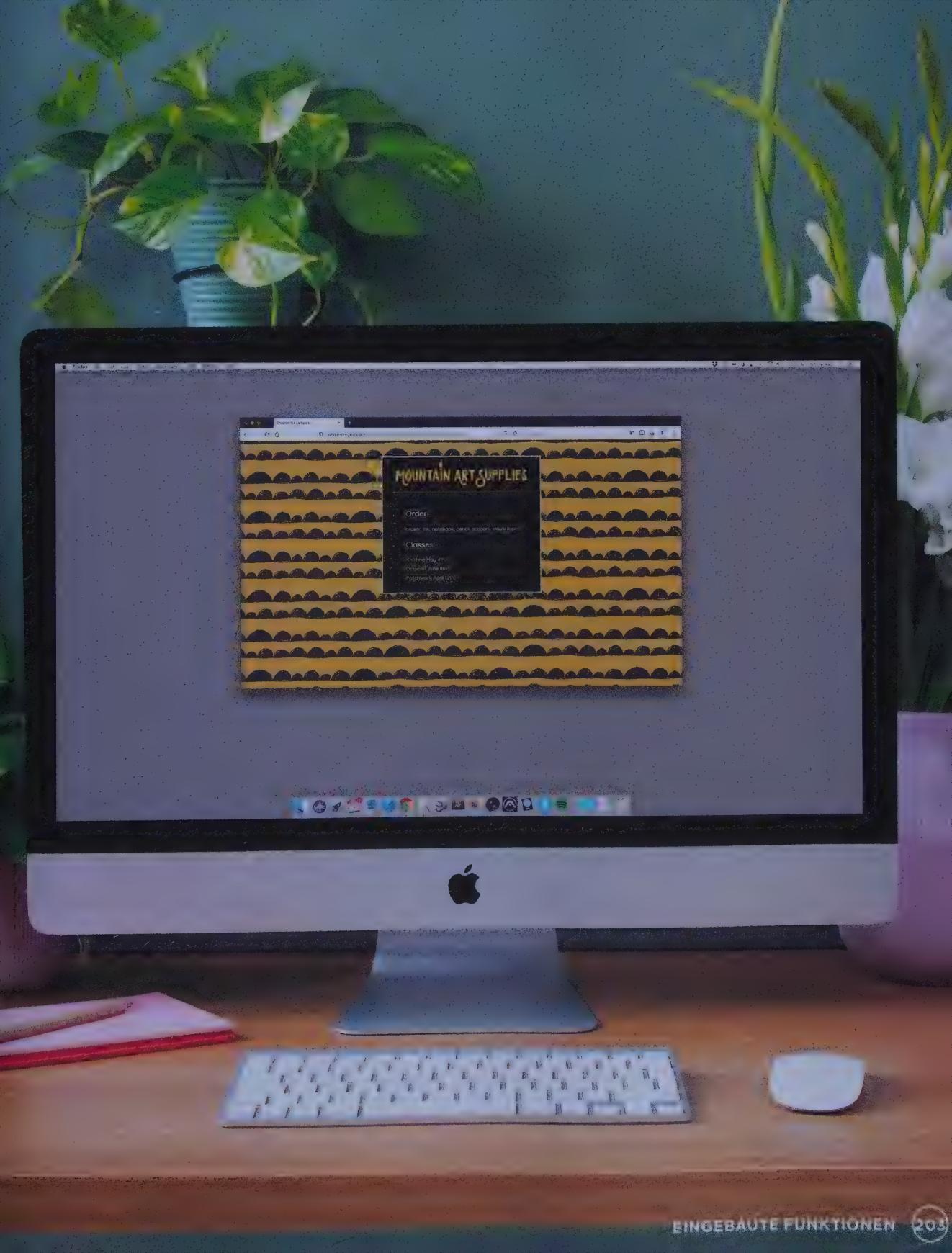
Die Definitionen der eingebauten Funktionen sind in den PHP-Interpreter integriert, müssen also vor dem Funktionsaufruf nicht in eine PHP-Seite eingebunden werden. Diese Funktionen erfüllen Aufgaben, die Webentwickler bei der Erstellung dynamischer Webseiten häufig benötigen, und sie ersparen es Ihnen, zu diesem Zweck eigene Funktionen schreiben zu müssen.

Um eine eingebaute Funktion aufzurufen, müssen Sie ihren Namen und ihre Parameter kennen und wissen, welche Daten Sie zurückgibt. Dieses Kapitel enthält daher mehrere Tabellen mit Funktionsnamen und Parametern sowie Beschreibungen der Funktionen und ihrer Rückgabewerte.

Die erste Auswahl an Funktionen ist nach den Datentypen gruppiert, die sie verarbeiten: Strings, Zahlen und Arrays. Weiter hinten in diesem Kapitel lernen Sie außerdem:

- wie Sie Konstante erstellen (vergleichbar mit Variablen, deren Werte nach ihrer Festlegung unverändert bleiben).
- eine Funktion kennen, mit der Sie steuern können, welche HTTP-Header vom PHP-Interpreter mit einer angeforderten Seite an den Browser zurückgesendet werden.
- eine Reihe von Funktionen kennen, mit denen Sie Informationen über Dateien auf dem Server abrufen können.

Die in diesem Kapitel behandelten Funktionen kommen im weiteren Verlauf dieses Buchs immer wieder zur Anwendung.



GROSS- & KLEINSCHREIBUNG, LÄNGE ERMITTELN

Diese Funktionen wandeln Text in Groß- oder Kleinbuchstaben um und zählen die Anzahl der Zeichen oder Wörter in einer Zeichenkette.

Die folgenden Funktionen sind für den Umgang mit Text (dem Datentyp String) konzipiert. Sie nehmen eine Zeichenkette als Argument entgegen, aktualisieren sie und geben die überarbeitete Zeichenkette zurück.

Die Funktion `strtolower()` nimmt beispielsweise eine Zeichenkette entgegen und wandelt den gesamten Text in Kleinbuchstaben um. Anschließend gibt sie den aktualisierten Wert zurück.

GROSS- UND KLEINSCHREIBUNG VERÄNDERN

FUNKTION	BESCHREIBUNG
<code>strtolower(\$string)</code>	Liefert eine Zeichenkette mit allen Zeichen in Kleinbuchstaben zurück.
<code>strtoupper(\$string)</code>	Liefert eine Zeichenkette mit allen Zeichen in Großbuchstaben zurück.
<code>ucwords(\$string)</code>	Gibt eine Zeichenkette zurück, bei der der erste Buchstabe jedes Worts ein Großbuchstabe ist.

ZEICHEN UND WÖRTER ZÄHLEN

FUNKTION	BESCHREIBUNG
<code>strlen(\$string)</code>	Gibt die Anzahl der Zeichen in der Zeichenkette zurück. Leerzeichen und Satzzeichen zählen als Zeichen. (Siehe auch <code>mb_strlen()</code> auf Seite 210–211 für Multibyte-Zeichen).
<code>str_word_count(\$string)</code>	Gibt die Anzahl der Wörter in der Zeichenkette zurück.

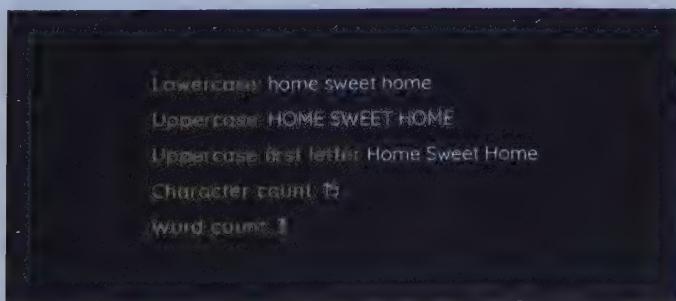
GROSS- & KLEINSCHREIBUNG VERÄNDERN & ZEICHEN ZÄHLEN

PHP

section_b/c05/case-and-character-count.php

```
<?php
① $text = 'Home sweet home';
?>
<?php include 'includes/header.php'; ?>
<p>
    <b>Lowercase:</b>
② <?= strtolower($text) ?><br>
    <b>Uppercase:</b>
③ <?= strtoupper($text) ?><br>
    <b>Uppercase first letter:</b>
④ <?= ucwords($text) ?><br>
    <b>Character count:</b>
⑤ <?= strlen($text) ?><br>
    <b>Word count:</b>
⑥ <?= str_word_count($text) ?>
</p>
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Die Zeichenkette

Home sweet home wird in der Variablen \$text gespeichert. Diese dient beim Aufruf der einzelnen Funktionen als Argument.

2. Die Funktion `strtolower()` wird aufgerufen. Die Funktion wandelt den Text in Kleinbuchstaben um und gibt diesen Wert zurück. Der Rückgabewert wird mit der Kurzschreibweise des echo-Befehls auf die Seite geschrieben.

3. Die Funktion `strtoupper()` liefert die Zeichenkette in Großbuchstaben zurück.

4. Die Funktion `ucwords()` liefert die Zeichenkette mit einem Großbuchstaben am Anfang jedes Worts zurück.

5. Die Funktion `strlen()` zählt die Zeichenanzahl im String und gibt diese Zahl zurück.

6. Die Funktion `str_word_count()` zählt die Wortanzahl im String und gibt diese Zahl zurück.

Probieren Sie es: Ändern Sie in Schritt 1 die Zeichenkette in PHP und MySQL, speichern Sie die Datei und aktualisieren Sie die Seite.

ZEICHEN IN EINER ZEICHENKETTE SUCHEN

Diese Funktionen suchen in einer Zeichenkette nach einem oder mehreren Zeichen. Wenn sie eine Übereinstimmung finden, geben sie die Position dieses Zeichens zurück. Wird keine Übereinstimmung gefunden, geben sie `false` zurück.

Jedem Zeichen in einer Zeichenkette ist eine **Position** zugeordnet, ein Zahlenwert, der bei 0 beginnt. Das erste Zeichen steht also an Position 0, das zweite an Position 1 und so weiter.

Home sweet home
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Wenn Sie innerhalb eines Strings nach einer bestimmten Zeichenfolge suchen, wird diese auch **Substring** genannt.

Einige der Funktionen unterscheiden zwischen **Groß- und Kleinschreibung**, sodass eine Übereinstimmung nur dann gefunden wird, wenn sowohl String als auch Substring die gleiche Kombination von Groß- und Kleinbuchstaben enthalten.

FUNKTION	BESCHREIBUNG
<code>strpos(\$string, \$substring[, \$offset])</code>	Gibt die Position der ersten Übereinstimmung für substring zurück (Groß-/Kleinschreibung wird beachtet). Bei Verwendung von offset wird nur hinter dieser Zeichenposition gesucht. wie <code>strpos()</code> , aber ohne Berücksichtigung der Groß-/Kleinschreibung.
<code>stripos(\$string, \$substring[, \$offset])</code>	Gibt die Position der letzten Übereinstimmung für substring zurück (Groß-/Kleinschreibung wird beachtet).
<code>strripos(\$string, \$substring[, \$offset])</code>	wie <code>strpos()</code> , aber ohne Berücksichtigung der Groß-/Kleinschreibung.
<code>strrstr(\$string, \$substring)</code>	Gibt den Text vom ersten Auftreten eines Substrings (einschließlich des Substrings) bis zum Ende der Zeichenkette zurück (Groß- und Kleinschreibung wird beachtet).
<code>stristr(\$string, \$substring)</code>	wie <code>strrstr()</code> , aber ohne Berücksichtigung der Groß-/Kleinschreibung.
<code>substr(\$string, \$offset[, \$characters])</code>	Gibt Zeichen ab der in \$offset angegebenen Position bis zu dem Ende der Zeichenkette zurück. (Alternativ gibt der optionale Parameter \$characters die Anzahl der Zeichen an, die nach \$offset zurückgegeben werden sollen.) Für weitere Optionen siehe: http://notes.re/php/substr .
<code>* str_contains(\$string, \$substring)</code>	Überprüft, ob ein Substring in einer Zeichenkette enthalten ist und liefert true/false zurück.
<code>* str_starts_with(\$string, \$substring)</code>	Überprüft, ob die Zeichenkette mit dem Substring beginnt, und gibt true/false zurück.
<code>* str_ends_with(\$string, \$substring)</code>	Überprüft, ob die Zeichenkette mit dem Substring endet, und gibt true/false zurück.

Die letzten drei mit Sternchen gekennzeichneten Funktionen wurden in PHP 8 hinzugefügt; sie unterscheiden alle zwischen Groß- und Kleinschreibung.

Hinweis: Optionale Parameter stehen in eckigen Klammern. Funktionen für Multibyte-Zeichen finden Sie auf Seite 210.

EINEN SUBSTRING SUCHEN

PHP

```
<?php  
① $text = 'Home sweet home';  
    ?> ...  
    <b>First match (case-sensitive):</b>  
② <?= strpos($text, 'ho') ?><br>  
    <b>First match (not case-sensitive):</b>  
③ <?= stripos($text, 'me', 5) ?><br>  
    <b>Last match (case-sensitive):</b>  
④ <?= strrpos($text, 'Ho') ?><br>  
    <b>Last match (not case-sensitive):</b>  
⑤ <?= stripos($text, 'Ho') ?><br>  
    <b>Text after first match (case-sensitive):</b>  
⑥ <?= strstr($text, 'ho') ?><br>  
    <b>Text after first match (not case-sensitive):</b>  
⑦ <?= stristr($text, 'ho') ?><br>  
    <b>Text between two positions:</b>  
⑧ <?= substr($text, 5, 5) ?><br>
```

ERGEBNIS

```
First match (case-sensitive):  
First match (not case-sensitive): 13  
Last match (case-sensitive): 0  
Last match (not case-sensitive): 11  
Text after first match (case-sensitive): home  
Text after first match (not case-sensitive): Home sweet home  
Text between two positions: home
```

Probieren Sie es: Ändern Sie in Schritt 1 die Zeichenkette in Home and family.

Verwenden Sie dann in Schritt 8 die Funktion substr(), um das Wort and zurückzuliefern.

1. Die Zeichenkette

Home sweet home wird in \$text gespeichert.

2. Die Funktion strpos() wird aufgerufen, um in der Zeichenkette nach dem ersten Auftreten des Substrings ho zu suchen. Der Rückgabewert ist 11.

3. Die Funktion stripos() wird aufgerufen, um das erste Auftreten des Substrings me nach der Position 5 zu finden. Sie liefert 13 zurück.

4. Die Funktion strrpos() wird aufgerufen, um das letzte Auftreten des Substrings Ho zu ermitteln. Sie liefert den Wert 0 zurück, da sie zwischen Groß- und Kleinschreibung unterscheidet.

5. Die Funktion stripos() wird aufgerufen, um das letzte Auftreten des Substrings Ho zu ermitteln. Sie liefert 11 zurück, da sie nicht zwischen Groß- und Kleinschreibung unterscheidet.

6. Die Funktion strstr() wird aufgerufen, um den Text ab dem ersten Auftreten des Substrings ho zu erhalten. Sie liefert home zurück.

7. Die Funktion stristr() wird aufgerufen, um sämtlichen Text ab dem ersten Auftreten von ho zurückzuliefern. Der Rückgabewert ist Home sweet home, da sie nicht zwischen Groß- und Kleinschreibung unterscheidet.

8. Die Funktion substr() wird aufgerufen und liefert fünf Zeichen zurück, beginnend mit dem Zeichen auf Position fünf.

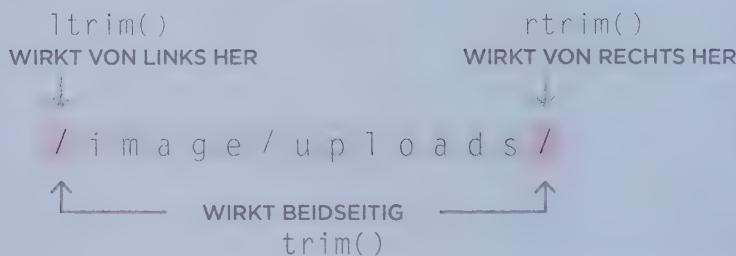
ZEICHEN ENTFERNEN UND ERSETZEN

Diese Funktionen können einzelne Zeichen (einschließlich Leerzeichen) entfernen, Zeichen ersetzen (wie ein Werkzeug zum Suchen und Ersetzen) und eine Zeichenkette eine festgelegte Anzahl von Malen wiederholen.

Die **trim**-Funktionen entfernen Zeichen aus einer Zeichenkette. Sie können nach bestimmten Zeichen am Anfang und/oder Ende der Zeichenkette suchen und diese löschen, wenn sie gefunden werden.

Wenn Sie keine zu löschen den Zeichen angeben, entfernen die trim-Funktionen alle Leerraumzeichen, die sich am Anfang und/oder Ende der Zeichenfolge befinden, dazu gehören Leerzeichen, Tabulatoren sowie harte und weiche Zeilenumbrüche.

Die **replace**-Funktionen suchen nach Zeichen in einer Zeichenkette. Falls eine Übereinstimmung besteht, werden diese Zeichen durch neue Zeichen ersetzt. Die Funktion **repeat** wiederholt einen String eine festgelegte Anzahl von Malen.



FUNKTION	BESCHREIBUNG
<code>ltrim(\$string[, \$delete])</code>	Entfernt Leerraum von der linken Seite der Zeichenkette. Der optionale Parameter <code>\$delete</code> legt einen Substring fest, der entfernt werden soll, falls er am Anfang der Zeichenkette auftritt. Groß- und Kleinschreibung werden dabei berücksichtigt.
<code>rtrim(\$string[, \$delete])</code>	Entfernt Leerraum von der rechten Seite der Zeichenkette.
<code>trim(\$string[, \$delete])</code>	Entfernt beidseitig eventuellen Leerraum von der Zeichenkette.
<code>str_replace(\$old, \$new, \$string)</code>	Ersetzt den Substring <code>\$old</code> durch denjenigen in <code>\$new</code> (Groß- und Kleinschreibung werden berücksichtigt).
<code>str_ireplace(\$old, \$new, \$string)</code>	Ersetzt den Substring <code>\$old</code> durch denjenigen in <code>\$new</code> (Groß- und Kleinschreibung werden ignoriert).
<code>str_repeat(\$string, \$repeats)</code>	Wiederholt die Zeichenkette eine bestimmte Anzahl von Malen.

ZEICHEN IN EINER ZEICHENKETTE ERSETZEN

PHP section_b/c05/removing-and-replacing-characters.php

```
<?php  
① $text = '/images/uploads/';  
?> ...  
② <?= trim($text, '/') ?><br>  
③ <?= ltrim($text, '/') ?><br>  
④ <?= rtrim($text, 's/') ?><br>  
⑤ <?= str_replace('images', 'img', $text) ?><br>  
⑥ <?= str_ireplace('IMAGES', 'img', $text) ?><br>  
⑦ <?= str_repeat($text, 2) ?></p>
```

ERGEBNIS

The screenshot shows a terminal window with the following text:

```
Remove '/' from both ends:  
images/uploads  
Remove ' ' from the left of the string:  
images/uploads/  
Remove 's/' from the right of the string:  
/images/upload  
Replace 'images' with 'img'  
/img/uploads/  
As above but case-insensitive:  
/img/uploads/  
Repeat the string:  
/images/uploads//images/uploads/
```

1. Der Pfad

/images/uploads/ wird als Zeichenkette in der Variablen \$text gespeichert.

2. Die Funktion trim()

entfernt links und rechts vom Text / und gibt die Zeichenkette zurück.

3. ltrim()

entfernt auf der linken Seite des Texts / und gibt die Zeichenkette zurück.

4. Die Funktion rtrim()

entfernt auf der rechten Seite des Texts / und gibt die Zeichenkette zurück.

5. str_replace()

gibt einen String zurück, in dem die Zeichenfolge images durch img ersetzt wurde. Groß- und Kleinschreibung werden dabei beachtet.

6. Die Funktion str_ireplace()

gibt einen String zurück, in der die Zeichenfolge IMAGES durch img ersetzt wurde. Die Substring-Suche unterscheidet nicht zwischen Groß- und Kleinschreibung, sodass sie sowohl IMAGES als auch images finden und durch img ersetzen würde.

7. Die Funktion str_repeat()

gibt die Zeichenkette mit allen Zeichen in zweifacher Ausfertigung zurück.

Probieren Sie es: Fügen Sie in Schritt 1 ein Leerzeichen vor und nach dem Dateipfad ein und aktualisieren Sie dann die Seite. Die Schrägstriche / werden in den Schritten 2, 3 oder 4 nicht mehr entfernt, da vor oder nach ihnen ein Leerzeichen steht.

MULTIBYTE-STRING-FUNKTIONEN

Einige der bisher vorgestellten String-Funktionen liefern bei Anwendung auf Multibyte-Zeichen ein falsches Ergebnis. Die folgenden Multibyte-String-Funktionen unterstützen alle Zeichen im UTF-8-Format.

Wenn Text mit UTF-8 kodiert wird, belegen einige Zeichen mehrere Datenbytes. So verwenden beispielsweise das £-Symbol zwei Bytes und das Euro-Symbol drei Bytes.

Wenn Sie Multibyte-Zeichen als Argumente für manche String-Funktionen verwenden, kann dies zu falschen Ergebnissen führen (Beispiele für fehlerhafte Ergebnisse finden Sie auf der rechten Seite).

Die unten gezeigten Multibyte-String-Funktionen tragen dieselben Namen wie die bisher in diesem Kapitel gezeigten Funktionen, jedoch wird ihnen die Zeichenfolge `mb_` vorangestellt.

Einige String-Funktionen haben keine Multibyte-Entsprechung, z.B. `trim()` und `str_replace()`. Sie funktionieren auch mit UTF-8, solange es als Standard-Zeichencodierung in einer der Dateien `php.ini` oder `.htaccess` festgelegt wurde.

FUNKTION	BESCHREIBUNG
<code>mb_strtoupper(\$string)</code>	Liefert die Zeichenkette komplett in Großbuchstaben zurück.
<code>mb_strtolower(\$string)</code>	Liefert die Zeichenkette komplett in Kleinbuchstaben zurück.
<code>mb_strlen(\$string)</code>	Liefert die Zeichenanzahl der Zeichenkette zurück.
<code>mb_strpos(\$string, \$substring[, \$offset])</code>	Gibt die Position des ersten Fundorts des Substrings zurück (unter Berücksichtigung der Groß- und Kleinschreibung). Wenn ein <code>\$offset</code> vorgegeben ist, wird nur nach dieser Zeichenposition gesucht.
<code>mb_stripos(\$string, \$substring[, \$offset])</code>	Version von <code>mb_strpos()</code> , die keine Groß- und Kleinschreibung berücksichtigt.
<code>mb_strrpos(\$string, \$substring[, \$offset])</code>	Liefert die Position des letzten Fundorts des Substrings zurück (Groß- und Kleinschreibung werden beachtet).
<code>mb_stripos(\$string, \$substring[, \$offset])</code>	Version von <code>mb_strrpos()</code> , die keine Groß- und Kleinschreibung berücksichtigt.
<code>mb_substr(\$string, \$start[, \$characters])</code>	Gibt den Text ab dem ersten Fundort eines Substrings (einschließlich dieses Substrings) bis zum Ende der Zeichenkette zurück (Groß- und Kleinschreibung werden beachtet).
<code>mb_substr(\$string, \$start[, \$characters])</code>	Version von <code>mb_substr()</code> , die keine Groß- und Kleinschreibung berücksichtigt.
<code>mb_stristr(\$string, \$substring)</code>	Gibt die Zeichen ab der in <code>\$start</code> angegebenen Position bis zum Ende der Zeichenkette zurück. Wird <code>\$characters</code> angegeben, liefert die Funktion diese Zeichenanzahl ab <code>\$start</code> zurück.

MULTIBYTE-STRING-FUNKTIONEN VERWENDEN

PHP

section_b/c05/multibyte-string-functions.php

```
<?php  
① $text = 'Total: £444';  
    ?> ...  
    <b>Character count using <code>strlen()</code>:</b>  
② <?= strlen($text) ?><br>  
    <b>Character count using <code>mb_strlen()</code>:</b>  
③ <?= mb_strlen($text) ?><br>  
    <b>First match of 444 <code>strpos()</code>:</b>  
④ <?= strpos($text, '444') ?><br>  
    <b>First match of 444 <code>mb_strpos()</code>:</b>  
⑤ <?= mb_strpos($text, '444') ?><br>
```

ERGEBNIS

```
Character count using strlen(): 12
Character count using mb_strlen(): 11
First match of 444 strpos(): 9
First match of 444 mb_strpos(): 8
```

Dieses Beispiel wendet String-Funktionen auf das £-Symbol an, dessen UTF-8-Kodierung zwei Bytes an Daten erfordert.

1. Eine Zeichenkette mit dem £-Symbol wird erstellt und in \$text gespeichert. Ihre Länge beträgt 11 Zeichen.
 2. Die Funktion `strlen()` zählt die Anzahl der Bytes, die zur Darstellung einer Zeichenkette benötigt werden, und nicht die Anzahl der Zeichen in der Zeichenkette. Deshalb meldet sie, dass die Zeichenkette 12 Zeichen enthält (statt 11).
 3. Die Funktion `mb_strlen()` berücksichtigt die vom PHP-Interpreter verwendete Kodierung und zeigt mit 11 die richtige Zeichenanzahl im String an.
 4. Die Funktion `strpos()` sucht das erste Auftreten der Zeichenfolge 444. Die Position wird anhand der Anzahl der Bytes ermittelt, die vor dem Fundort des Substrings stehen (und nicht anhand der Zeichenanzahl). Sie gibt daher 9 und nicht 8 zurück.
 5. `mb_strpos()` ermittelt die erste Position von 444 und gibt korrekt den Wert 8 zurück.
- Probieren Sie es:** Tauschen Sie in der Zeichenkette in Schritt 1 das £-Symbol gegen das Euro-Symbol aus.

REGULÄRE AUSDRÜCKE

Kreditkartennummern, Postleitzahlen und Telefonnummern bilden bestimmte Zeichenmuster. Reguläre Ausdrücke beschreiben ein Muster aus Zeichen, und PHP verfügt über eingebaute Funktionen, um zu prüfen, ob diese Muster in einer Zeichenkette auftauchen.

Reguläre Ausdrücke stehen zwischen zwei Schrägstrichen. Das Muster im folgenden Ausdruck beschreibt:

[A-z] die Buchstaben A-z (Groß- und Kleinbuchstaben)

{3,9} mit einer Anzahl zwischen 3 und 9 Stück

Würde dieser Ausdruck zur Überprüfung der unten stehenden Zeichenkette verwendet, würde der PHP-Interpreter die **erste** Zeichenfolge finden, die aus 3-9 der Groß- oder Kleinbuchstaben A-z besteht. Diese Zeichen sind darunter hervorgehoben:

Thomas was 1st!

Thom^as was 1st !

/[A-z]{3,9}/

Die Syntax für reguläre Ausdrücke kann recht komplex sein, und es gibt ganze Bücher darüber, wie man sie schreibt, aber diese Seiten vermitteln Ihnen die Grundlagen.

Die folgende Tabelle zeigt, wie Sie bestimmten Zeichen, einem bestimmten Zeichenbereich und Zeichen am Anfang oder Ende einer Zeichenfolge entsprechen können.

AUSDRUCK	BESCHREIBUNG	BEISPIEL
/1st/	Entspricht den Zeichen 1st.	Thomas was 1st!
/[ab-cde]/	Für Zeichen in eckigen Klammern führt ein beliebiger dieser Buchstaben zur Übereinstimmung; hier gilt dies für die Buchstaben a, b, c, d oder e	Thom ^a s was 1st !
/[K-Z]/	Ein Trennstrich in eckigen Klammern erzeugt einen Zeichenbereich. Auf diesen Ausdruck passen alle Großbuchstaben zwischen K und Z.	Thomas was 1st!
/[a-e]/	Dies entspricht allen Kleinbuchstaben von a bis e.	Thomas was 1st!
/[0-9]/	Dies entspricht allen Ziffern von 0 bis 9.	Thomas was 1st!
/[A-z0-9]/	Dies entspricht allen Klein- oder Großbuchstaben von A bis z oder den Ziffern 0 bis 9.	Thomas was 1st!
/^A-Z/	Ein Zirkumflex ^ am Anfang eines Musters besagt, dass die Zeichenkette mit diesen Zeichen beginnen muss; hier gibt es also nur eine Übereinstimmung, wenn das erste Zeichen aus dem Bereich A-Z kommt.	Thomas was 1st !
/1st\!\$/	Ein Dollar \$ am Ende eines Musters besagt, dass die Zeichenkette mit dieser Zeichenfolge enden muss; hier passt dies, wenn die letzten Zeichen 1st sind!	Thomas was 1st !
/\s/	Entspricht einem Leerzeichen.	Thom ^a s' wa ^s ' 1st !

Die nachfolgenden Zeichen haben in regulären Ausdrücken eine besondere Bedeutung: \ / . | \$ () ^ ? { } + *

Um ein Muster zu erstellen, das auf eines dieser Zeichen passt, stellen Sie einen **Backslash** vor das Zeichen.

AUSDRUCK	BESCHREIBUNG	BEISPIEL
/[\!?\(\)]/	Entspricht einem Ausrufezeichen, Fragezeichen oder einer Klammer.	Thomas was 1st!

Sie können einen **Quantor** hinzufügen, um anzugeben, wie oft ein Muster in der Zeichenkette vorkommen soll.

In den folgenden Beispielen werden Zeichen gesucht, die mit einer bestimmten Häufigkeit vorkommen.

AUSDRUCK	BESCHREIBUNG	BEISPIEL
/[a-z]+/	Das Pluszeichen + steht für eines oder mehrere der angegebenen Zeichen.	Thomas was 1st!
/[a-z]{3}/	Eine Zahl in geschweiften Klammern {} gibt an, wie oft das Muster genau gefunden werden muss.	Thomas was 1st!
/[A-z]{3,5}/	Zwei durch ein Komma getrennte Zahlen in geschweiften Klammern {} geben an, wie oft das Muster mindestens und höchstens gefunden werden muss.	Thomas was 1st!
/[a-z]{3,}/	Eine Zahl gefolgt von einem Komma (ohne zweite Zahl) in geschweiften Klammern gibt an, wie oft das Muster mindestens gefunden werden muss.	Thomas was 1st!

Um nach einer Reihe von Mustern zu suchen, geben Sie die Suchmuster nacheinander ein.

Im Folgenden muss auf eine Übereinstimmung mit dem ersten Muster eine Übereinstimmung mit dem zweiten Muster folgen.

AUSDRUCK	BESCHREIBUNG	BEISPIEL
/[0-9][a-z]/	Entspricht einer Ziffer 0-9 gefolgt von einem Kleinbuchstaben a - z.	Thomas was 1st!

Wenn Sie einen Ausdruck teilweise in Klammern setzen, entsteht eine Gruppe. Mit einem Quantor hinter der Gruppe können Sie angeben, wie oft sie erscheinen soll.

Wenn Sie eine von mehreren Optionen suchen wollen, können Sie die Optionen in einer Gruppe durch ein Pipe-Symbol getrennt angeben.

AUSDRUCK	BESCHREIBUNG	BEISPIEL
/[0-9]([a-z]{2})/	[0-9] entspricht einer Ziffer 0-9; darauf folgt ([a-z]{2}), was zwei Kleinbuchstaben entspricht.	Thomas was 1st!
/[1-31](st nd r-d th)/	[1-31] passt auf eine Zahl von 1-31 gefolgt von (st nd rd th), was auf st, nd, rd oder th passt.	Thomas was 1st!

FUNKTIONEN FÜR REGULÄRE AUSDRÜCKE

Diese Funktionen prüfen, ob eine Zeichenkette ein Zeichenmuster enthält, das durch einen regulären Ausdruck beschrieben wird. Wird eine Übereinstimmung gefunden, führen die Funktionen jeweils unterschiedliche Aufgaben aus.

Die folgenden Funktionen verwenden alle regulären Ausdrücke, um in einer Zeichenkette nach einem bestimmten Zeichenmuster zu suchen.

Sie erledigen Aufgaben wie:

- Überprüfen, ob das Muster gefunden wird
- Zählen, wie oft ein Muster gefunden wird
- das Zeichenmuster suchen und durch eine neue Zeichenfolge ersetzen (wie die Funktion Suchen/Ersetzen in einem Textverarbeitungsprogramm)

Diese Funktionen haben Parameter für:

- den reguläreren Ausdruck, der das gesuchte Zeichenmuster beschreibt (in Anführungszeichen, da es sich um eine Zeichenkette handelt)
- den String, in dem nach diesem Zeichenmuster gesucht wird

Die Funktion, die ein passendes Zeichenmuster durch einen neuen Substring ersetzt, muss auch wissen, wodurch diese Zeichen ersetzt werden sollen.

FUNKTION	BESCHREIBUNG
<code>preg_match(\$regex, \$string)</code>	Sucht in einer Zeichenkette nach einem passenden Muster. Der Rückgabewert ist 1, wenn eine Übereinstimmung gefunden wurde, 0, wenn keine Übereinstimmung gefunden wurde, und <code>false</code> , wenn ein Fehler aufgetreten ist.
<code>preg_match_all(\$regex, \$string)</code>	Sucht in einer Zeichenkette nach einem passenden Muster und gibt die Anzahl der gefundenen Übereinstimmungen zurück (0, wenn keine gefunden wurde) oder <code>false</code> , wenn ein Fehler aufgetreten ist.
<code>preg_split(\$regex, \$string)</code>	Sucht in einer Zeichenkette nach einem passenden Muster. Jedes Mal, wenn eine Übereinstimmung gefunden wird, wird die Zeichenkette geteilt und jeder dieser Teile in einem indizierten Array zurückgegeben.
<code>preg_replace(\$regex, \$replace, \$string)</code>	Ersetzt die angegebenen Zeichen durch eine alternative Zeichenfolge, ähnlich wie beim Suchen und Ersetzen in einer Textverarbeitung. Gibt den String mit den ersetzen Zeichen zurück oder <code>null</code> , falls ein Fehler aufgetreten ist. Zum Löschen von Zeichen ersetzen Sie sie durch einen leeren String.

Hinweis: Diese Funktionsnamen beginnen mit dem Präfix preg (für Perl regular expressions), da die in PHP verwendeten regulären Ausdrücke denen einer anderen Programmiersprache namens Perl gleichen.

REGULÄRE AUSDRÜCKE VERWENDEN

PHP

section_b/c05/regular-expression-functions.php

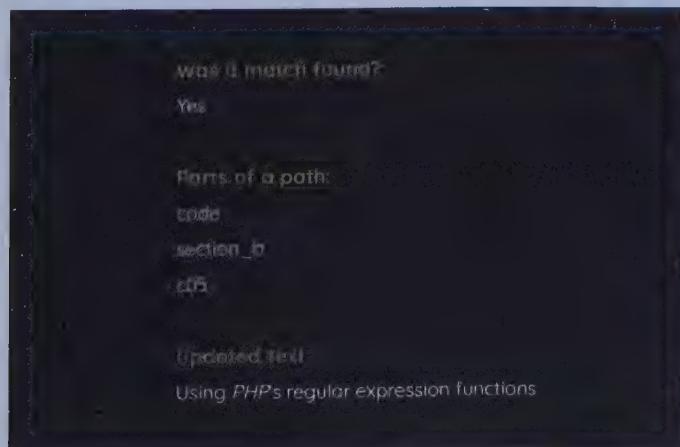
```
<?php
① $text = 'Using PHP\'s regular expression functions';
② $path = 'code/section_b/c05/';

③ $match = preg_match('/PHP/', $text);
④ $path  = preg_split('/\//', $path);
⑤ $text  = preg_replace('/PHP/', '<em>PHP</em>', $text);
?> ...
<b>Was a match found?</b><br>
⑥ <?= ($match == 1) ? 'Yes' : 'No' ?><br><br>

<b>Parts of a path:</b><br>
⑦ <?php foreach($path as $part) { ?>
    <?= $part ?><br>
<?php } ?>

<b>Updated text:</b><br>
⑧ <?= $text ?>
```

ERGEBNIS



1. Text wird in der Variablen \$text gespeichert.
2. In der Variablen \$path wird ein Dateipfad gespeichert.
3. Die Funktion preg_match() prüft, ob die Zeichenkette PHP in dem Text gefunden wird, der in Schritt 1 in der Variablen \$text gespeichert wurde. Wird sie gefunden, erhält \$match den Wert 1.
4. Die Funktion preg_split() teilt den in Schritt 2 in der Variablen \$path gespeicherten Pfad bei jedem Schrägstrich auf und fügt die einzelnen Teile in ein neues Element eines Arrays ein.
5. Die Funktion preg_replace() sucht in dem Text, der in Schritt 1 in \$text gespeichert wurde, nach der Buchstabenfolge PHP. Wird diese gefunden, so wird sie durch dieselben Buchstaben in einem -HTML-Element ersetzt.
6. Mit einem ternären Operator wird überprüft, ob die Variable \$match den Wert 1 enthält. Trifft dies zu, wird das Wort Yes auf der Seite ausgegeben. Andernfalls wird stattdessen das Wort No angezeigt.
7. Eine Schleife gibt jedes Element im Array \$path auf einer neuen Zeile aus.
8. Der aktualisierte Text in \$text wird angezeigt, wobei die Zeichenfolge PHP in -Tags steht.

MIT ZAHLEN ARBEITEN

Zusätzlich zu den bereits in Kapitel 1 vorgestellten mathematischen Operatoren gibt es Funktionen für häufig durchgeführte Vorgänge mit Zahlen.

FUNKTION	BESCHREIBUNG
<code>round(\$number, \$places, \$round)</code>	Rundet Fließkommazahlen auf oder ab: \$number ist die Zahl, die auf- oder abgerundet werden soll. \$places ist die Anzahl der Dezimalstellen, auf die gerundet werden soll. \$round gibt an, wie Zahlen auf- oder abgerundet werden sollen:
OPTION	ZWECK
<code>PHP_ROUND_HALF_UP</code>	Rundet Halbe auf (aus 3.5 wird z. B. 4).
<code>PHP_ROUND_HALF_DOWN</code>	Rundet Halbe ab (aus 3.5 wird z. B. 3).
<code>PHP_ROUND_HALF_EVEN</code>	Rundet Halbe auf die nächste gerade Zahl.
<code>PHP_ROUND_HALF_ODD</code>	Rundet Halbe auf die nächste ungerade Zahl.
<code>ceil(\$number)</code>	Rundet eine Zahl auf die nächste Ganzzahl auf.
<code>floor(\$number)</code>	Rundet eine Zahl auf die nächste Ganzzahl ab.
<code>mt_rand(\$min, \$max)</code>	Erzeugt eine Pseudozufallszahl zwischen \$min und \$max.
<code>rand(\$min, \$max)</code>	Seit PHP 7.1 sind rand() und mt_rand() identisch. Davor verwendete rand() einen Algorithmus, der weniger zufällig und langsamer war.
<code>pow(\$base, \$exponent)</code>	Potenziert die Basis mit dem Exponenten und liefert den entsprechenden Rückgabewert (z. B. 3 ⁴ würde 81 liefern).
<code>sqrt(\$number)</code>	Liefert die Quadratwurzel einer Zahl zurück.
<code>is_numeric(\$number)</code>	Überprüft, ob ein Wert eine Zahl ist (entweder Ganz- oder Fließkommazahl). Liefert true zurück, falls es sich um eine Zahl handelt, ansonsten false.
<code>number_format(\$number [, \$decimals] [, \$decimal_point] [, \$thousand_separator])</code>	Gibt an, wie eine Zahl formatiert werden soll. Wenn nur \$number angegeben wird, wird die Zahl ohne Dezimalstellen formatiert, und ein Komma dient zur Trennung von Tausenderstellen. \$decimals stellt die angegebene Zahl von Dezimalstellen dar, wobei diese mit einem Punkt und die Tausender mit einem Komma abgetrennt werden. Mit \$decimal_point und \$thousand_separator können Sie die Zeichen angeben, die zur Trennung von Dezial- und Tausenderstellen verwendet werden. Um decimal_point oder thousand_separator einzusetzen, müssen Sie immer beide gleichzeitig verwenden.

NUMERISCHE FUNKTIONEN

1. Zahlen werden auf unterschiedliche Weise gerundet.
2. Es wird eine Zufallszahl zwischen 0 und 10 erzeugt.
3. 4 hoch 5 wird angezeigt.

4. Die Quadratwurzel aus 16 wird angezeigt.
5. Prüft, ob ein Wert eine Zahl ist (vom Datentyp int oder float). Gibt in diesem Fall true zurück (Ausgabe auf der Seite ist 1); ansonsten false (keine Ausgabe).

6. Die Zahl wird mit 2 Dezimalstellen formatiert. Ein Leerzeichen trennt die Tausender- und ein Komma die Dezimalstellen ab.

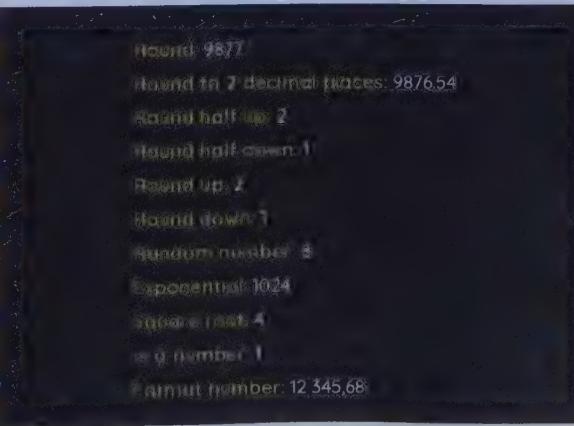
Probieren Sie es: Erzeugen Sie in Schritt 2 eine Zufallszahl zwischen 50 und 100.

PHP

section_b/c05/numerical-functions.php

```
① <b>Round:</b>                      <?= round(9876.54321) ?><br>
    <b>Round to 2 decimal places:</b> <?= round(9876.54321, 2) ?><br>
    <b>Round half up:</b>              <?= round(1.5, 0, PHP_ROUND_HALF_UP) ?><br>
    <b>Round half down:</b>            <?= round(1.5, 0, PHP_ROUND_HALF_DOWN) ?><br>
    <b>Round up:</b>                  <?= ceil(1.23) ?><br>
    <b>Round down:</b>                <?= floor(1.23) ?><br>
② <b>Random number:</b>                <?= mt_rand(0, 10) ?><br>
③ <b>Exponential:</b>                  <?= pow(4, 5) ?><br>
④ <b>Square root:</b>                 <?= sqrt(16) ?><br>
⑤ <b>Is a number:</b>                  <?= is_numeric(123) ?><br>
⑥ <b>Format number:</b>                <?= number_format(12345.6789, 2, ',', ',') ?><br>
```

ERGEBNIS



MIT ARRAYS ARBEITEN

Diese Funktionen können den Inhalt eines Arrays durchsuchen, die Anzahl der darin enthaltenen Elemente ermitteln und zufällige Schlüssel aus dem Array auswählen. Sie können ein Array auch in eine Zeichenkette umwandeln oder umgekehrt.

Wie wir gesehen haben, enthalten Arrays mehrere Schlüssel/Wertpaare in einer einzigen Variablen.

In einem indizierten Array ist der Schlüssel eine Indexnummer. Er gibt die Position des Elements im Array an.

Ein assoziatives Array gleicht eher einer Sammlung von zusammenhängenden Variablen. Als Schlüssel dient dort jeweils eine Zeichenkette.

INFORMATIONEN ÜBER EIN ARRAY ERHALTEN

FUNKTION	BESCHREIBUNG
<code>array_key_exists(\$key, \$array)</code>	Überprüft, ob es einen bestimmten Schlüssel im Array gibt. Gibt true zurück, wenn er existiert, ansonsten false.
<code>array_search(\$value, \$array[, \$strict])</code>	Durchsucht die im Array gespeicherten Werte und gibt den Schlüssel für die erste Übereinstimmung zurück. Wenn \$strict auf true gesetzt ist, muss für eine Übereinstimmung auch der Datentyp passen.
<code>in_array(\$value, \$array)</code>	Prüft, ob ein Wert in einem Array enthalten ist. Gibt true zurück, wenn ja, ansonsten false.
<code>count(\$array)</code>	Gibt die Anzahl der Elemente im Array zurück.
<code>array_rand(\$array[, \$number])</code>	Wählt ein zufälliges Element aus dem Array aus und gibt dessen Schlüssel zurück. Wird im zweiten Parameter eine Zahl angegeben, ist der Rückgabewert ein Array mit dieser Anzahl zufällig ausgewählter Schlüssel aus dem ersten Array.

ARRAYS IN ZEICHENKETTEN UMWANDELN UND UMGEGEHRT

FUNKTION	BESCHREIBUNG
<code>implode([\$separator], \$array)</code>	Wandelt die Werte eines Arrays in eine Zeichenkette um (Schlüssel werden weggelassen). Wenn Sie ein Trennzeichen (\$separator) angeben, wird es zwischen den einzelnen Werten eingefügt.
<code>explode(\$separator, \$string[, \$limit])</code>	Wandelt eine Zeichenkette in ein indiziertes Array um. Der Separator ist das Zeichen, das die einzelnen Elemente der Zeichenkette voneinander trennt. Mit dem Limit können Sie bei Bedarf die maximale Anzahl der Elemente festlegen, die dem Array hinzugefügt werden.

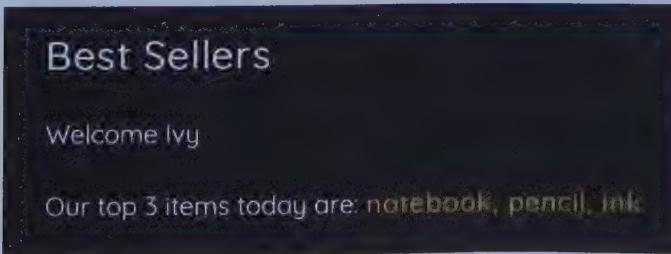
ARRAY-FUNKTIONEN

PHP

section_b/c05/array-functions.php

```
<?php
    // Begrüßungs-Array erstellen, Zufallswert auswählen
    $greetings = ['Hi ', 'Howdy ', 'Hello ', 'Hola ',
    'Welcome ', 'Ciao ',];
    $greeting_key = array_rand($greetings);
    $greeting = $greetings[$greeting_key];
    // Bestseller-Array, Artikel zählen, Top-Artikel
    $bestsellers = ['notebook', 'pencil', 'ink'];
    $bestseller_count = count($bestsellers);
    $bestseller_text = implode(', ', $bestsellers);
    // Array mit Kundendaten
    $customer = ['forename' => 'Ivy',
    'surname' => 'Stone',
    'email' => 'ivy@eg.link'];
    // Wenn Kundenvorname, zur Begrüßung hinzufügen
    if (array_key_exists('forename', $customer)) {
        $greeting .= $customer['forename'];
    }
    ?> ...
    <h1>Best Sellers</h1>
    <p><?= $greeting ?></p>
    <p>Our top <?= $bestseller_count ?> items today are:
    <b><?= $bestseller_text ?></b></p>
```

ERGEBNIS



1. Im Array `$greetings` werden mehrere Grußformeln abgelegt.
2. Ein zufälliger Schlüssel wird aus dem Array ausgewählt und in der Variablen `$greeting_key` gespeichert.
3. Mit dem Zufallschlüssel wird die Begrüßung aus dem Array abgerufen und in `$greeting` gespeichert.
4. Ein Array der meistverkauften Artikel wird in `$bestsellers` gespeichert.
5. Der Funktion `count()` zählt die Elemente dieses Arrays; die ermittelte Anzahl wird in `$bestseller_count` gespeichert.
6. Das Array wird mit `implode()` in die Zeichenkette `$bestseller_text` umgewandelt, wobei die einzelnen Elemente durch ein Komma voneinander getrennt werden.
7. Ein assoziatives Array mit Kundeninformationen wird erstellt.
8. `array_key_exists()` prüft, ob es einen Wert für den Vornamen gibt. Ist dies der Fall, wird er an `$greeting` angehängt.
9. Die Begrüßung wird ausgegeben.
10. Die Anzahl der Bestseller und deren Artikelbezeichnungen werden ausgegeben.

Probieren Sie es: Ergänzen Sie in Schritt 4 die Bestsellerliste um einen weiteren Artikel.

ARRAY-ELEMENTE HINZUFÜGEN UND ENTFERNEN

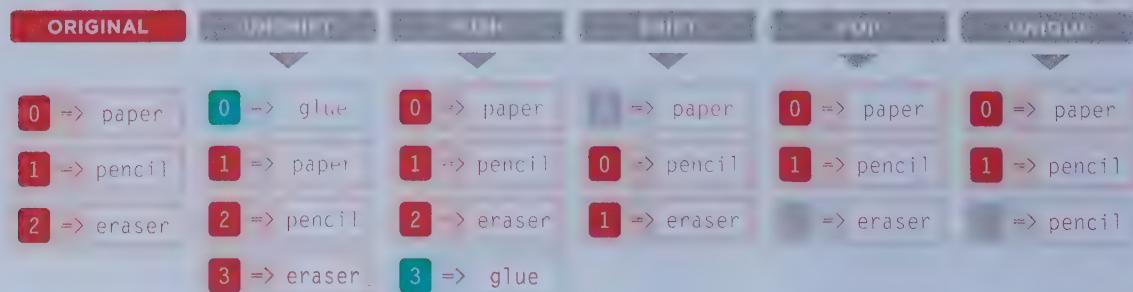
Diese Funktionen fügen einem Array Elemente hinzu oder entfernen sie daraus. Sie können festlegen, ob die neuen Elemente am Anfang oder Ende des Arrays angefügt werden sollen.

Um ein Element in ein Array aufzunehmen, geben Sie den neuen Wert an.

Um ein Element aus einem Array zu entfernen, geben Sie lediglich seinen Schlüssel an.

Das Diagramm zeigt die Positionen der hinzugefügten oder entfernten Elemente.

FUNKTION	BESCHREIBUNG
<code>array_unshift(\$array, \$items)</code>	Fügt ein oder mehrere Elemente am Anfang eines indizierten Arrays ein. Gibt die Anzahl der Elemente in dem Array zurück. (Mehr über assoziative Arrays erfahren Sie auf Seite 42).
<code>array_push(\$array, \$items)</code>	Fügt ein oder mehrere Elemente am Ende eines indizierten Arrays ein. Gibt die Anzahl der Elemente in dem Array zurück. (Mehr über assoziative Arrays erfahren Sie auf Seite 42).
<code>array_shift(\$array)</code>	Entfernt das erste Element aus dem Array. Gibt den Wert des entfernten Elements zurück.
<code>array_pop(\$array)</code>	Entfernt das letzte Element aus dem Array. Gibt den Wert des entfernten Elements zurück.
<code>array_unique(\$array)</code>	Entfernt doppelte Einträge aus einem Array und liefert das neue Array zurück.
<code>array_merge(\$array1, \$array2)</code>	Verbindet zwei oder mehr Arrays und liefert das neue Array zurück. Handelt es sich um zwei indizierte Arrays, beginnen die Indexnummern des neuen Arrays bei 0. Sie können zwei Arrays auch mit dem Operator + verbinden: \$array1 + \$array2.



FUNKTIONEN ZUR AKTUALISIERUNG VON ARRAYS

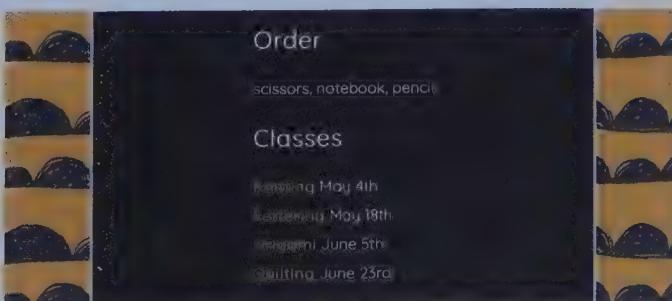
PHP

section_b/c05/array-updating-functions.php

```
<?php
    // Array der bestellten Artikel
    ① $order = ['notebook', 'pencil', 'eraser',];
    ② array_unshift($order, 'scissors'); // vorne anfügen
    ③ array_pop($order);             // Letztes entfernen
    ④ $items = implode(', ', $order); // in String umwandeln

    // Array der angebotenen Kurse
    ⑤ $classes = ['Patchwork' => 'April 12th',
                  'Knitting'  => 'May 4th',
                  'Lettering' => 'May 18th',];
    ⑥ array_shift($classes);           // Erstes entfernen
    ⑦ $new      = ['Origami'  => 'June 5th',
                  'Quilting'   => 'June 23rd',];
    ⑧ $classes = array_merge($classes, $new); // hinten anfügen
    ?>
    <h1>Order</h1>
    ⑨ <?= $items ?>
    <h1>Classes</h1>
    ⑩ <?php foreach($classes as $description => $date) { ?>
        <b><?= $description ?></b> <?= $date ?><br>
    <?php } ?>
```

ERGEBNIS



1. Das indiziertes Array \$order wird erstellt.

2. Die Funktion array_unshift() fügt ein Element am Anfang des Arrays an. Der erste Parameter ist das Array; der zweite das anzufügende Element (das funktioniert so nur bei indizierten Arrays).

3. Die Funktion array_pop() entfernt das letzte Element.

4. Das Array wird mit implode() in eine Zeichenkette umgewandelt und in \$items gespeichert. Die einzelnen Elemente werden durch ein Komma und ein Leerzeichen voneinander getrennt.

5. Das assoziative Array \$classes wird erstellt.

6. Die Funktion array_shift() entfernt das erste Element aus dem Array.

7. Ein weiteres assoziatives Array wird erstellt, um neue Elemente aufzunehmen.

8. Mittels array_merge() wird das Array \$classes um die in Schritt 7 neu erstellten Elemente ergänzt.

9. \$items wird ausgegeben.

10. Eine foreach-Schleife gibt die Schlüssel und Werte des assoziativen Arrays aus.

Probieren Sie es: Trennen Sie in Schritt 4 die Elemente in der Zeichenkette durch ein Semikolon.

ARRAYS SORTIEREN (REIHENFOLGE ÄNDERN)

Die Sortierfunktionen ändern die Reihenfolge der Array-Elemente. In aufsteigende Listen sind die Elemente vom niedrigsten zum höchsten Wert geordnet (z.B. A-Z oder 0-9). In absteigenden Listen sind die Elemente vom höchsten zum niedrigsten Wert geordnet (z.B. Z-A oder 9-0).

NACH WERT SORTIEREN UND SCHLÜSSEL ÄNDERN

Wenn Sie das Array mit den nachfolgenden Funktionen sortieren, werden die Schlüssel zu Indexnummern, die bei 0 beginnen (egal ob es sich um ein indiziertes oder assoziatives Array handelt).

Das `r` in `rsort()` steht für reverse (rückwärts).

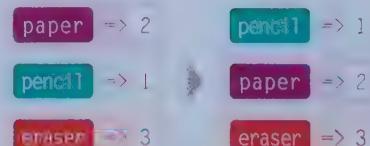
FUNKTION	BESCHREIBUNG
<code>sort(\$array)</code>	nach aufsteigendem Wert
<code>rsort(\$array)</code>	nach absteigendem Wert



NACH WERT SORTIEREN UND SCHLÜSSEL BEIBEHALTEN

Wenn Sie das Array mit den nachfolgenden Funktionen sortieren, werden die Schlüssel gemeinsam mit ihren Werten verschoben.

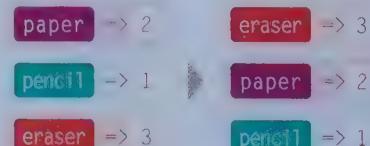
FUNKTION	BESCHREIBUNG
<code>asort(\$array)</code>	nach aufsteigendem Wert
<code>arsort(\$array)</code>	nach absteigendem Wert



NACH SCHLÜSSEL SORTIEREN UND WERTE BEIBEHALTEN

Wenn Sie das Array mit den nachfolgenden Funktionen sortieren, werden die Werte gemeinsam mit ihren Schlüsseln verschoben.

FUNKTION	BESCHREIBUNG
<code>ksort(\$array)</code>	nach aufsteigendem Schlüssel
<code>krsort(\$array)</code>	nach absteigendem Schlüssel



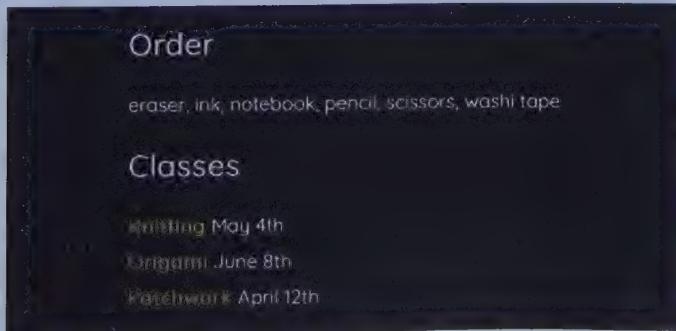
ARRAY-SORTIERFUNKTIONEN

PHP

section_b/c05/array-sorting-functions.php

```
<?php  
    // Array der bestellten Artikel  
    ① $order = ['notebook', 'pencil', 'scissors',  
               'eraser', 'ink', 'washi tape',];  
    ② sort($order);                      // aufsteigend sortieren  
    ③ $items = implode(', ', $order);   // in Text umwandeln  
  
    // Array mit Kursangeboten anlegen  
    ④ $classes = ['Patchwork' => 'April 12th',  
                  'Knitting'  => 'May 4th',  
                  'Origami'   => 'June 8th',];  
    ⑤ ksort($classes);                 // nach Schlüssel sortieren  
    ?>  
  
<h1>Order</h1>  
⑥ <?= $items ?>  
<h1>Classes</h1>  
⑦ <?php foreach($classes as $description => $date) { ?>  
    <b><?= $description ?></b> <?= $date ?><br>  
<?php } ?>
```

ERGEBNIS



1. Ein indiziertes Array wird erstellt und in der Variablen \$order gespeichert.
2. Die Werte im Array werden mit sort() in aufsteigender alphabetischer Reihenfolge sortiert. Dadurch erhält jedes Element im Array eine neue Indexnummer, beginnend bei 0.
3. Das Array wird mit implode() in eine Zeichenkette umgewandelt. Die einzelnen Elemente werden durch ein Komma und ein Leerzeichen voneinander getrennt. Die resultierende Zeichenkette wird in der Variablen \$items gespeichert.
4. Ein assoziatives Array wird erstellt und in der Variablen \$classes gespeichert.
5. Die Schlüssel im Array werden mittels ksort() in alphabetischer Reihenfolge sortiert (ihre Werte werden mit verschoben).
6. Die in \$items gespeicherte Zeichenkette wird ausgegeben.
7. Mit einer foreach-Schleife werden die Schlüssel und Werte des Arrays \$classes auf der Seite ausgegeben.

Probieren Sie es: Kehren Sie in Schritt 5 die Reihenfolge des Arrays \$classes um.

KONSTANTE

Eine Konstante ist ein Name/Wert-Paar, das sich wie eine Variable verhält. Allerdings kann ihr nur einmal ein Wert zugewiesen werden, der sich dann nicht mehr aktualisieren lässt.

Eine Konstante ist ein Name/Wert-Paar wie eine Variable, aber:

- Sie wird mit der Funktion `define()` erstellt.
- Ihr Wert kann, einmal gesetzt, nicht mehr verändert werden.
- Der Zugriff auf die Konstante ist überall auf der PHP-Seite möglich (auch innerhalb von Funktionen).

Der Name der Konstanten sollte die Art der darin enthaltenen Daten beschreiben und mit einem Buchstaben oder Unterstrich beginnen (kein Dollarzeichen). Als Werte sind skalare Datentypen oder Arrays möglich.

Die Parameter der Funktion `define()` sind:

- Der Name der Konstanten, der normalerweise aus Großbuchstaben besteht.
- Ihr Wert; Zeichenketten sollten in Anführungszeichen stehen, Zahlen und boolesche Werte nicht.
- Ein optionaler boolescher Wert, der angibt, ob beim Namen zwischen Groß- und Kleinschreibung unterschieden wird oder nicht (`true`, wenn ja, `false`, wenn nicht). Wird der dritte Parameter nicht angegeben, gilt für den Namen eine Unterscheidung zwischen Groß- und Kleinschreibung.

```
define('SITE_NAME', 'Mountain Art Supplies');
```



Konstanten werden häufig genutzt, um Informationen zu speichern, die eine Website für ihre Funktion benötigt, deren Werte sich aber nur ändern, wenn eine Website installiert wird (erstmalig, beim Umzug auf einen neuen Server oder wenn dieselbe Code für eine andere Website verwendet wird).

Eine Konstante kann auch mit dem Schlüsselwort `const` erstellt werden, gefolgt vom Namen der Konstanten, dem Zuweisungsoperator und dem gewünschten Wert. Mit dieser Vorgehensweise kann eine Konstante innerhalb einer Klasse definiert werden (was mit der Funktion `define()` nicht möglich ist).

```
const SITE_NAME = 'Mountain Art Supplies';
```



KONSTANTEN VERWENDEN

PHP

section_b/c05/includes/settings.php

```
<?php  
① define('SITE_NAME', 'Mountain Art Supplies');  
② const ADMIN_EMAIL = 'admin@eg.link';
```

PHP

section_b/c05/includes/constants.php

```
<?php  
③ include 'includes/settings.php';  
include 'includes/header.php';  
?  
④ <h1>Welcome to <?= SITE_NAME ?></h1>  
⑤ <p>To contact us, email <?= ADMIN_EMAIL ?></p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



In diesem Beispiel legt die Include-Datei settings.php zwei Konstanten an, die Informationen über die Website enthalten.

1. Mit der Funktion `define()` wird die Konstante `SITE_NAME` angelegt. Ihr Wert gibt den Namen der Website an.

2. Mit dem Schlüsselwort `const` wird die Konstante `ADMIN_EMAIL` angelegt. Ihr Wert entspricht der E-Mail-Adresse des Website-Betreibers.

Die zweite Datei in diesem Beispiel ist die Seite constants.php, die auf die Werte dieser beiden Konstanten zurückgreift.

3. Die Datei settings.php wird eingebunden, damit die Seite auf die Konstanten zugreifen kann.

4. Mit der Kurzschreibweise des echo-Befehls wird der Inhalt der Konstanten, die den Websitenamen enthält, ausgegeben.

5. Die E-Mail-Adresse des Website-Betreibers wird angezeigt.

HTTP-HEADER HINZUFÜGEN ODER AKTUALISIEREN

Die Funktion `header()` aktualisiert die vom PHP-Interpreter an den Browser gesendeten HTTP-Header. Sie kann auch neue Header hinzufügen. Als einziges Argument nimmt sie den Namen des zu setzenden Headers, gefolgt von einem Doppelpunkt und dessen Wert, entgegen.

Manchmal müssen Sie Benutzer nach einem Seitenaufruf auf eine andere Seite weiterleiten. Zum Beispiel, wenn die ursprünglich angeforderte Seite:

- nicht mehr verfügbar ist
- auf eine neue URL umgezogen ist
- nicht über alle benötigten Daten verfügt

In diesem Fall besitzt `header()` ein Argument, das sich aus drei Teilen zusammensetzt:

- die Header-Bezeichnung `Location`
- ein Doppelpunkt
- die neue URL

Wenn der Browser den Location-Header empfängt, fordert er die neue URL an. Anschließend sollte der Befehl `exit` folgen, damit der PHP-Interpreter keinen weiteren Code ausführt (siehe rechts).

`header('Location: http://www.example.com/');`

HEADER NEUE URL

Die meisten PHP-Dateien erzeugen HTML, das an den Browser übermittelt wird, aber mit PHP lassen sich auch andere Dateitypen wie JSON, XML oder CSS generieren.

Hierzu benötigt `header()`:

- den Header `Content-type`
- einen Doppelpunkt
- den Medientyp des neuen Inhalts

Dadurch wird ein HTTP-Header erzeugt, der dem Browser den Medientyp der Datei mitteilt. Weitere Informationen zu Medientypen finden Sie hier: <http://notes.re/media-types>.

`header('Content-type: application/json');`

HEADER MEDIENTYP TYPE

Browser können besuchte Seiten im Cache zwischenspeichern. Fordert eine Benutzerin die Seite erneut an, kann er die gespeicherte Seite anzeigen, statt die Datei erneut vom Server anzufordern (dies scheint die Ladezeit zu verkürzen).

Um dem Browser mitzuteilen, für welchen Zeitraum er eine Seite zwischenspeichern kann, verwenden Sie:

- den Header `Cache-Control`
- einen Doppelpunkt
- `max-age=` gefolgt von der Anzahl an Millisekunden, die die Seite zwischengespeichert werden darf

Internetanbieter und Netzwerke nutzen Proxys, um Webseiten zwischenzuspeichern. Für Seiten mit personenbezogenen Daten setzen Sie hinter die Millisekunden ein Komma, ein Leerzeichen und das Wort `private`, um die Zwischenspeicherung durch einen Proxy zu verhindern. Enthält die Seite keine persönlichen Daten, setzen Sie den Wert auf `public`.

`header('Cache-Control: max-age=3600, public');`

HEADER GÜLTIG FÜR ZEIT (MS) PROXY

NUTZERANFRAGEN MIT HTTP-HEADERN UMLEITEN

PHP

section_b/c05/redirect.php

```
<?php  
① $logged_in = true;  
  
② if ($logged_in == false) {  
③     header('Location: login.php');  
④     exit;  
}  
?>  
<?php include 'includes/header.php'; ?>  
<h1>Members Area</h1>  
<p>Welcome to the members area</p>  
<?php include 'includes/footer.php'; ?>
```

PHP

section_b/c05/login.php

```
<h1>Login</h1>  
<b>You need to log in to view this page.</b>  
<p>(You create a full login system in Chapter 16.)</p>
```

ERGEBNIS



Dieses Beispiel zeigt, wie Sie Benutzer mit der Funktion `header()` auf eine andere Seite umleiten können. Vor dem Einsatz der `header()`-Funktion dürfen Sie keinerlei Markup oder Text an den Browser gesendet haben, nicht einmal ein einzelnes Leerzeichen oder einen Zeilenumbruch.

1. Die Variable `$logged_in` speichert einen booleschen Wert, der angibt, ob der Benutzer eingeloggt ist.

2. In einer `if`-Anweisung wird mit einer Bedingung überprüft, ob der Wert in `$logged_in` false entspricht.

3. Entspricht der Wert false, wird der Benutzer mithilfe der Funktion `header()` auf die Seite `login.php` umgeleitet.

(Wie Sie einen Mitgliederbereich mit einer funktionierenden Login-Seite erstellen, erfahren Sie in Kapitel 16).

4. Nach der Umleitung mit der Funktion `header()` dient der Befehl `exit` dazu, die weitere Ausführung des PHP-Codes in der Datei zu unterbinden.

Enthält `$logged_in` den Wert true, wurde der vorhergehende Codeblock übersprungen, und der Rest der Seite wird angezeigt.

Probieren Sie es: Ändern Sie in Schritt 1 den Wert der Variablen `$logged_in` auf false. Sie werden dann zur Anmeldeseite weitergeleitet.

INFORMATIONEN ÜBER DATEIEN UND DATEIEN LÖSCHEN

Die Dateifunktionen akzeptieren als Parameter einen Dateipfad und liefern dann entweder Informationen über die Datei und ihren Pfad zurück oder löschen die Datei.

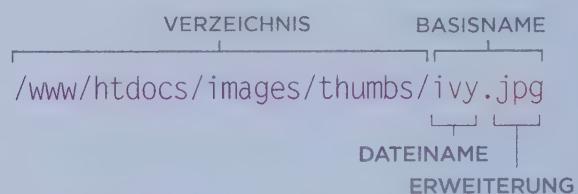
Die nachfolgende Tabelle listet Dateifunktionen auf.

Einige dieser Funktionen liefern verschiedene Teile eines Dateipfads zurück, die im Diagramm rechts veranschaulicht werden.

PHP verfügt auch über eingebaute Konstanten, die Pfade speichern:

`__FILE__` enthält den Pfad der aktuellen Datei.

`__DIR__` enthält das Verzeichnis der aktuellen Datei.



FUNKTION	BESCHREIBUNG
<code>file_exists(\$path)</code>	Überprüft, ob eine Datei existiert. Gibt <code>true</code> zurück, wenn sie existiert, ansonsten <code>false</code> .
<code>filesize(\$path)</code>	Gibt die Dateigröße in Bytes zurück.
<code>mime_content_type(\$path)</code>	Gibt den Medientyp der Datei zurück (siehe http://notes.re/media-types).
<code>unlink(\$path)</code>	Versucht, eine Datei zu löschen, und liefert im Erfolgsfall <code>true</code> zurück, ansonsten <code>false</code> .
<code>pathinfo(\$path[, \$part])</code>	Gibt Teile des Dateipfads zurück. Sie können angeben, welcher Teil des Pfads abgerufen werden soll. Tun Sie dies nicht, wird ein Array mit den folgenden vier Schlüsseln zurückgeliefert:
TEIL	BESCHREIBUNG
<code>PATHINFO_DIRNAME</code>	Pfad zum Verzeichnis, in dem die Datei liegt.
<code>PATHINFO_BASENAME</code>	Dateiname inklusiver Erweiterung (basename).
<code>PATHINFO_FILENAME</code>	Dateiname (ohne Erweiterung).
<code>PATHINFO_EXTENSION</code>	Dateinamenerweiterung.
<code>basename(\$path)</code>	Liefert aus einem Pfad den vollständigen Dateinamen (inklusiver Erweiterung) zurück.
<code>dirname(\$path[, \$levels])</code>	Liefert den Pfad zu dem Verzeichnis zurück, in dem sich die angegebene Datei befindet. Wenn <code>\$levels</code> angegeben wird, entspricht dies der Anzahl der übergeordneten Verzeichnisse, die mit ausgegeben werden.
<code>realpath(\$path)</code>	Liefert den absoluten Dateipfad zurück.

Der Unterschied zwischen absoluten und relativen Pfaden wird hier beschrieben: <http://notes.re/paths>.

DATEIINFORMATIONEN ERHALTEN

PHP

section_b/c05/files.php

```
<?php  
① $path = 'img/logo.png';  
?>  
<?php include 'includes/header.php'; ?>  
② <?php if (file_exists($path)) { ?>  
③   <b>Name:</b>      <?= pathinfo($path, PATHINFO_BASENAME) ?><br>  
④   <b>Size:</b>       <?= filesize($path) ?> bytes<br>  
⑤   <b>Mime type:</b> <?= mime_content_type($path) ?><br>  
⑥   <b>Folder:</b>     <?= pathinfo($path, PATHINFO_DIRNAME) ?><br>  
<?php } else { ?>  
⑦   <p>There is no such file.</p>  
<?php } ?>  
<?php include 'includes/footer.php'; ??
```

ERGEBNIS



Probieren Sie es: Ändern Sie \$path in Schritt 1 in img/pattern.png. Sie sehen den neuen Namen und die neue Dateigröße (der Mime-Typ und der Ordner bleiben gleich).

Probieren Sie es: Ändern Sie \$path in Schritt 1 in img/nologo.png. Da diese Datei nicht existiert, sollten Sie die Fehlermeldung aus Schritt 7 erhalten.

1. \$path speichert einen Dateipfad.
2. Eine if-Anweisung überprüft anhand von file_exists(), ob die Datei existiert. Falls ja, werden Informationen über die Datei ausgegeben.
3. pathinfo() gibt den Namen der Datei einschließlich ihrer Erweiterung (den sogenannten basename) aus.
4. filesize() zeigt die Dateigröße in Bytes an.
5. mime_content_type() zeigt den Medientyp der Datei an.
6. pathinfo() zeigt an, in welchem Verzeichnis sich die Datei befindet.
7. Wenn die Datei nicht existiert, erfolgt die Meldung, dass es keine solche Datei gibt.

ZUSAMMENFASSUNG

EINGEBAUTE FUNKTIONEN

- › Die in PHP integrierten Funktionen übernehmen Aufgaben, die bei der Erstellung von Websites häufig benötigt werden.
- › Eingebaute Funktionen rufen Sie genau wie jede andere Funktion auf, fügen dazu aber keine Funktionsdefinition in die Seite ein.
- › String-Funktionen finden, zählen und ersetzen Zeichen oder verändern deren Groß- und Kleinschreibung.
- › Numerische Funktionen runden Werte, erzeugen Zufallszahlen und führen mathematische Operationen durch.
- › Array-Funktionen ergänzen und entfernen Elemente, sortieren den Inhalt eines Arrays, suchen nach Schlüsseln oder Werten und verwandeln Arrays in Zeichenketten und umgekehrt.
- › Konstante sind wie Variablen, deren Wert jedoch nicht mehr geändert werden kann, sobald er einmal festgelegt wurde.
- › Die Funktion `header()` aktualisiert die an den Browser gesendeten HTTP-Header (und kann Benutzer auf eine andere Seite umleiten).

6

DATEN VON
BROWSERN
ERHALTEN

In diesem Kapitel lernen Sie, auf die Daten zuzugreifen, die der Browser an den PHP-Interpreter sendet, und sicherzustellen, dass diese für die Anzeige auf dynamischen Webseiten nutzbar und sicher sind..

In der Einführung zu diesem Buchteil haben Sie erfahren, dass HTML-Seiten über zwei Mechanismen verfügen, um Daten an den Server zu senden: Informationen an Links anhängen oder Formulare zum Ausfüllen bereitzustellen. Sie haben auch gesehen, wie diese Daten über HTTP GET (in einem Query-String) oder HTTP POST (in den HTTP-Headern, die mit jeder Seitenanfrage mitgesendet werden) übermittelt werden.

In diesem Kapitel lernen Sie, wie Sie Zugang zu diesen Daten erhalten, damit sie auf der Seite Verwendung finden können. Dies beinhaltet vier wichtige Schritte:

- Die einzelnen Datenfragmente aus dem Query-String oder den HTTP-Headern **extrahieren**.
- Die einzelnen Teildaten **validieren**, um zu überprüfen, ob ein Wert angegeben wurde und ob dieser im richtigen Format vorliegt (wenn eine Seite z.B. eine Zahl benötigt, wird geprüft, ob eine Zahl eingegeben wurde und kein Text).
- **Entscheiden**, ob die Seite die vom Besucher eingegebenen Daten verarbeiten kann oder nicht. Falls nicht, müssen eventuell entsprechende Fehlermeldungen angezeigt werden.
- Die Daten so **bereinigen**, dass ihre Verwendung auf der Seite kein Sicherheitsrisiko darstellt. Bestimmte Zeichen können die korrekte Anzeige einer Seite verhindern oder der Website sogar schaden.

Für diese vier Schritte gibt es keine Standardverfahren; verschiedene Entwicklerinnen und Entwickler verfolgen unterschiedliche Ansätze. In diesem Kapitel wird eine Auswahl der vielen verschiedenen Möglichkeiten vorgestellt, Daten einzuholen und zu gewährleisten, sodass sie sicher genutzt werden können.



VIER SCHRITTE ZUM SAMMELN UND NUTZEN VON DATEN

Um Besucherdaten zu erfassen und ihren sicheren Gebrauch zu gewährleisten benötigen Sie vier Schritte:

1. DATEN ERFASSEN

Zunächst erfassen Sie die vom Browser an den Server gesendeten Daten. Dazu können Sie Folgendes verwenden:

- Zwei superglobale Arrays, die der PHP-Interpreter bei jeder Anforderung einer Datei erstellt.
- Zwei eingebaute sogenannte **Filterfunktionen**.

Sie werden sehen, dass eine Seite nicht immer alle zur Ausführung ihrer Aufgabe erforderlichen Werte erhält, was einen Fehler verursachen kann.

Wenn eine Teilinformation optional ist, können Sie ihr einen Standardwert zuweisen, auf den die Seite zurückgreifen soll, falls sie nicht angegeben wird.

Wenn eine Angabe nicht optional ist und Daten fehlen, müssen Sie dem Besucher gegebenenfalls mitteilen, dass er keine ausreichenden Angaben gemacht hat (siehe nächster Schritt).

2. DATEN ÜBERPRÜFEN

Nachdem eine PHP-Seite Daten von einem Browser eingeholt hat, **validiert** sie häufig jedes einzelne dieser Datenelemente, um sicherzustellen, dass bei der Ausführung der Seite keine Fehler dadurch auftreten. Dabei wird geprüft:

- Ob die Seite die zur Ausführung ihrer Aufgabe benötigten Daten hat. Dies sind also die **erforderlichen Daten** oder Pflichtangaben.
- Ob die Daten im **richtigen Format** vorliegen. Wenn Ihre Seite zur Durchführung einer Berechnung beispielsweise eine Zahl benötigt, können Sie überprüfen, ob Sie eine Zahl bekommen haben. Oder Sie erwarten eine E-Mail-Adresse und prüfen, ob der Text das richtige Format für eine gültige E-Mail-Adresse besitzt.

PHP bietet zwei Möglichkeiten zur Validierung von Daten. Sie können:

- Ihre eigenen benutzerdefinierten Funktionen schreiben.
- Mit den Filterfunktionen eine Reihe von eingebauten Filtern verwenden. Jeder Filter validiert verschiedene Arten von Daten.

Es gibt verschiedene Möglichkeiten, diese einzelnen Schritte umzusetzen, und in diesem Kapitel lernen Sie einige davon kennen.

3. MASSNAHME BESCHLIESSEN

Nachdem eine Seite alle benötigten Einzelwerte zusammengetragen und validiert hat, kann sie überprüfen, ob alle für die Ausführung erforderlichen Daten vorliegen oder nicht:

- Wenn alle Daten gültig sind, können sie verarbeitet werden.
- Sind einige der Daten ungültig oder fehlen, sollten sie nicht verwendet werden. Stattdessen kann die Seite eine Fehlermeldung ausgeben.

Das Vorgehen zur Anzeige von Fehlern wegen ungültiger Daten ist bei Formularen etwas anders als bei Query-Strings.

- Bei ungültigen Formulardaten können Sie das Formular erneut anzeigen und dabei neben den Formularsteuerelementen, die ungültige Daten geliefert haben, entsprechende Meldungen einblenden. Die Meldungen sollten den Benutzer darauf hinweisen, wie er die Daten im korrekten Format eingeben kann.
- Wenn ein Query-String fehlerhafte Daten enthält, sollten Sie nicht erwarten, dass die Benutzer den Query-String bearbeiten. Stattdessen sollten Sie eine Meldung bereitstellen, die erklärt, wie der Benutzer die gewünschten Daten abfragen kann.

4. DATEN BEREINIGEN

Immer wenn Sie Daten verarbeiten, die ein Besucher auf einer Seite eingegeben hat, müssen diese **maskiert** werden, um sicherzustellen, dass sie sicher angezeigt werden können. Dazu gehört es, einige Zeichen, die von Browern als Code behandelt werden (z. B. die Symbole < und >), durch sogenannte HTML-Entitäten zu ersetzen. Die **Entitäten** weisen den Browser an, diese Zeichen anzuzeigen (anstatt sie als HTML-Code auszuführen).

Wenn Sie diesen Schritt nicht durchführen, bevor Sie die Daten auf einer Seite anzeigen, könnte eine Hackerrin versuchen, die Seite zur Ausführung einer bösartigen JavaScript-Datei zu veranlassen.

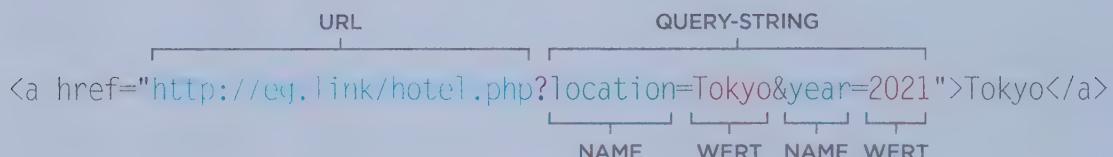
Wenn der Benutzer Daten angibt, die später in einer URL verwendet werden, müssen Sie auch spezielle Sonderzeichen wie etwa Schrägstriche und Fragezeichen maskieren. Wenn Sie dies nicht tun, kann der Webserver die URL möglicherweise nicht verarbeiten.

DATEN ÜBER HTTP GET VERSENDEN

Wenn Daten in einen Query-String am Ende einer URL eingefügt werden, lädt der PHP-Interpreter diese Daten in das superglobale Array `$_GET`, damit der PHP-Code der Seite darauf zugreifen kann.

Unten sehen Sie einen HTML-Link. In seinem href-Attribut steht die URL der verlinkten Seite.

Am Ende der URL befindet sich ein Query-String mit zwei Name/Wert-Paaren, die an den Server gesendet werden, sobald der Link angeklickt wird.

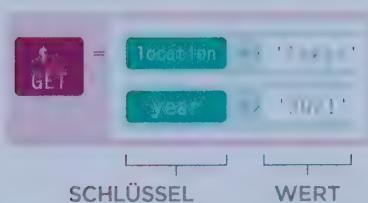


Wenn der PHP-Interpreter diese Anfrage erhält, fügt er die Daten aus dem Query-String in das superglobale Array `$_GET` ein. Wie alle vom PHP-Interpreter erzeugten superglobalen Arrays ist auch `$_GET` ein assoziatives Array. Für jedes Name/Wert-Paar im Abfrage-String wird darin ein Element angelegt:

- Der **Schlüssel** ist der gesendete Name.
 - Der **Wert** ist der mit dem Namen mitgesendete Wert.

Der Code in der PHP-Datei kann nach demselben Muster auf die Werte im superglobalen Array `$_GET` zugreifen wie auf die Werte eines beliebigen anderen assoziativen Arrays:

```
$location = $_GET['location'];
```



Oft werden mehrere Seiten einer Website mit einer einzigen PHP-Datei angezeigt, und anhand der Daten im Query-String wird festgelegt, welche Daten auf der Seite dargestellt werden.

Auf der rechten Seite gibt es ein Array mit drei Elementen. Jedes Element enthält die Stadt und die Adresse eines Geschäfts. Ein Wert im Query-String wählt aus, welche Daten welches Geschäfts angezeigt werden sollen, sodass diese eine PHP-Datei mit drei Seiten der Website generiert; für jedes der Geschäfte eine eigene.

Die Daten im Array dienen auch zur Erstellung der Links, mit denen diese drei Seiten aufgerufen werden.

INHALTE MITHILFE EINES QUERY-STRINGS AUSWÄHLEN

PHP

section_b/c06/get-1.php?city=London

```
<?php
$cities = [
    'London' => '48 Store Street, WC1E 7BS',
    'Sydney' => '151 Oxford Street, 2021',
    'NYC'      => '1242 7th Street, 10492',
];
① $city    = $_GET['city'];
③ $address = $cities[$city];
?>
...
④ <?php foreach ($cities as $key => $value) { ?>
⑤     <a href="get-1.php?city=<?= $key ?>"><?= $key ?></a>
<?php } ?>

⑥ <h1><?= $city ?></h1>
<p><?= $address ?></p>
```

ERGEBNIS



Probieren Sie es: Entfernen Sie in der Adressleiste des Browsers den Query-String aus der URL und laden Sie die Seite neu. Es werden zwei Fehler angezeigt.

Dies liegt daran, dass der Städtename nicht im Query-String enthalten war und daher nicht in das superglobale Array `$_GET` übernommen wurde.

In diesem Beispiel wird der Name einer Stadt aus dem Query-String erfasst und die Adresse eines Geschäfts in dieser Stadt angezeigt.

1. Die Variable `$cities` enthält ein assoziatives Array. Die einzelnen Schlüssel sind unterschiedliche Städtenamen; die zugehörigen Werte enthalten die Adressen der Filialen in den jeweiligen Städten.
2. Der Name der Stadt wird aus dem superglobalen Array `$_GET` abgerufen und in der Variablen `$city` gespeichert. (Achten Sie hier bitte auf die Groß- und Klein-schreibung.)
3. Der Städtename wird zur Auswahl der Filialadresse in dieser Stadt aus dem in Schritt 1 erstellten Array genutzt, die dann in der Variablen `$address` abgelegt wird.
4. Mit einer `foreach`-Schleife wird jedes Element der `$cities`-Arrays durchlaufen.
5. Innerhalb der Schleife wird für jede Stadt ein Link erzeugt. Der Name der Stadt wird im Query-String und nochmals als Linktext ausgegeben. Dies verdeutlicht, wie PHP Links erstellen kann und wie diese Links auf eine einzige Datei verweisen können, die unterschiedliche Daten anzeigt.
6. In den Schritten 2 und 3 in den Variablen `$city` und `$address` gespeicherten Werte werden auf der Seite angezeigt.

MIT FEHLENDEN DATEN IN SUPERGLOBALEN ARRAYS UMGEHEN

Wenn Sie versuchen, auf einen Schlüssel zuzugreifen, der nicht zu einem superglobalen Array hinzugefügt wurde, gibt der PHP-Interpreter einen Fehler aus. Zur Vermeidung derartiger Fehler können Sie vor dem Zugriff auf den Schlüssel kontrollieren, ob dieser im superglobalen Array enthalten ist.

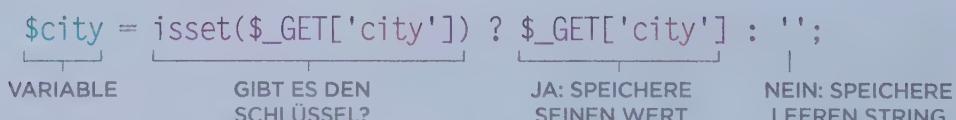
Gibt jemand den Link zu einer Seite weiter, kann es passieren, dass der Query-String versehentlich ganz oder teilweise fehlt.

Am Ende des vorherigen Beispiels haben Sie gesehen, dass fehlende Query-String-Daten natürlich auch nicht in das superglobale Array `$_GET` aufgenommen werden können. Versucht eine PHP-Datei dann, auf diese Daten zuzugreifen, reagiert der PHP-Interpreter mit der Fehlermeldung `Undefined array key` oder `Undefined index`, weil er versucht, auf einen Schlüssel (oder Index) zuzugreifen, der nicht zum superglobalen Array `$_GET` hinzugefügt wurde.

Um diesen Fehler zu vermeiden, sollten Seiten vor dem Zugriff auf das superglobale Array `$_GET` prüfen, ob diesem ein bestimmter Wert hinzugefügt wurde.

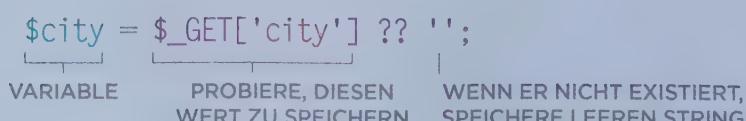
PHP verfügt über die eingebaute Funktion `isset()`, die einen Variablenamen, einen Array-Schlüssel oder eine Objekteigenschaft als Argument akzeptiert. Sie liefert `true` zurück, wenn die Variable, der Schlüssel oder die Eigenschaft existiert und ihr Wert nicht `null` ist. Andernfalls gibt sie `false` zurück, ohne dass jedoch ein Fehler auftritt.

Unten wird die Variable `$city` deklariert. Ein ternärer Operator (die Kurzform einer `if ... else`-Anweisung, siehe die Seiten 76 und 77) prüft, ob die superglobale Variable `$_GET` einen Schlüssel `city` enthält und ob dessen Wert nicht `null` ist. Trifft dies zu, wird er in der Variablen `$city` gespeichert. Andernfalls wird `$city` mit einer leeren Zeichenkette gefüllt.



Mit PHP 7 wurde der Null-Koaleszenz-Operator eingeführt, der als Kurzschreibweise für den Einsatz von `isset()` in der Bedingung eines ternären Operators dient.

Für den Fall, dass der Wert links vom Null-Koaleszenz-Operator nicht existiert, geben Sie rechts davon einen Alternativwert an, der dann verwendet werden soll.



EINEN QUERY-STRING ZUR AUSWAHL VON INHALTEN VERWENDEN

PHP

section_p/c06/get-2.php

```
<?php
$cities = [
    'London' => '48 Store Street, WC1E 7BS',
    'Sydney' => '151 Oxford Street, 2021',
    'NYC'     => '1242 7th Street, 10492',
];
① $city = $_GET['city'] ?? '';
② if ($city) {
③     $address = $cities[$city];
④ } else {
⑤     $address = 'Please select a city';
}
?>
...
<?php foreach ($cities as $key => $value) { ?>
    <a href="get-2.php?city=<?= $key ?>"><?= $key ?></a>
<?php } ?>

<h1><?= $city ?></h1>
<p><?= $address ?></p>
```

ERGEBNIS



Dieses Beispiel baut auf dem vorhergehenden auf. Die Unterschiede wurden hervorgehoben.

1. Der in der Variablen `$city` gespeicherte Wert wird mithilfe des Null-Koaleszenz-Operators zugewiesen. Wenn das superglobale Array `$_GET`:

- den Schlüssel `city` enthält, dessen Wert nicht `null` ist, wird der Wert in der Variablen `$city` gespeichert.
- keinen Schlüssel `city` enthält oder dessen Wert `null` ist, wird ein leerer String in der Variablen `$city` gespeichert.

2. Die Variable `$city` wird in der Bedingung einer `if`-Anweisung eingesetzt. Wenn der Wert ein String ist, der nicht leer ist, behandelt der PHP-Interpreter ihn als wahr und führt den nachfolgenden Code-Block aus.

3. Die Variable `$address` speichert die Adresse der Filiale in der im Query-String genannten Stadt.

4. Andernfalls (wenn der Wert in der Variablen `$city` ein leerer String ist) wird der zweite Code-Block ausgeführt.

5. In der Variablen `$address` wird eine Nachricht gespeichert, die den Benutzer zur Auswahl einer Stadt auffordert.

Probieren Sie es: Geben Sie im Query-String Tokyo als Stadt an. Die Seite gibt einen Fehler aus, weil sie diesen Schlüssel im Array `$cities` nicht finden kann. Fügen Sie dem Array in Schritt 1 ein neues Element mit dem Schlüssel Tokyo hinzu und ergänzen Sie es um eine Adresse. Probieren Sie anschließend erneut, diesen Städtenamen im Query-String zu verwenden.

DATENVALIDIERUNG

Bevor eine PHP-Seite die abgerufenen Daten verwendet, sollten diese auf Plausibilität geprüft werden, damit sie bei der Ausführung der Seite keine Fehler verursachen.

Bei der Validierung der Daten, die eine Seite empfängt, wird geprüft, ob der PHP-Code:

- die zur Ausführung einer Aufgabe benötigten Daten (Pflichteingaben) vorliegen.
- die Daten im richtigen Format vorliegen. Wenn eine Seite z.B. eine Zahl für eine Berechnung benötigt, können Sie prüfen, ob sie eine Zahl (und keine Zeichenkette) erhalten hat.

In der Datei auf der letzten Seite musste der Query-String Folgendes enthalten:

- Name und Wert zur Bestimmung der anzuseigenen Filiale
- Wert, der mit einem Schlüssel des Städte-Arrays übereinstimmt

Wenn der im Query-String angegebene Wert nicht im Array `$cities` enthalten ist, gibt der PHP-Interpreter einen Fehler aus. Der Code kann also, bevor er versucht, die Stadt auf der Seite auszugeben, prüfen, ob der Wert im Query-String im Array `$cities` zu finden ist.

Im weiteren Verlauf dieses Kapitels werden Sie verschiedene Wege kennenlernen, um verschiedene Arten von Daten zu validieren.

Das Beispiel rechts überprüft mit der in PHP eingebauten Funktion `array_key_exists()` (siehe Seite 218), ob der Wert im Query-String mit einem Schlüssel im Array `$cities` übereinstimmt. Die Funktion gibt `true` zurück, wenn der Schlüssel gefunden wurde, und `false`, wenn nicht. Der Rückgabewert der Funktion wird in der Variablen `$valid` gespeichert.

Nach der Datenprüfung muss die Seite entscheiden, ob sie den restlichen Code ausführen soll oder nicht.



Das bereits zuvor im Kapitel verwendete Beispiel wird auf der rechten Seite ausgebaut. Mit der Variablen `$valid` wird in der Bedingung einer `if`-Anweisung ermittelt, ob die Seite die Daten verarbeiten kann oder nicht:

- Sind die Daten gültig, kann die Seite die Adresse der Filiale aus dem Array abrufen und in der Variablen `$address` speichern, um sie später auf der Seite anzeigen zu können.
- Sind die Daten nicht valide, wird in der Variablen `$address` eine Nachricht abgelegt, die zur Auswahl einer Stadt auffordert. Dies liefert eine hilfreiche Rückmeldung zur Nutzung der Seite und zum Abrufen der gesuchten Informationen.

Im weiteren Verlauf des Kapitels lernen Sie mit Seiten umzugehen, die mehrere Werte vom Browser abfragen müssen, und wie Sie überprüfen, ob all diese Werte gültig sind oder nicht.

QUERY-STRING DATEN ÜBERPRÜFEN

PHP

section_b/c06/get-3.php

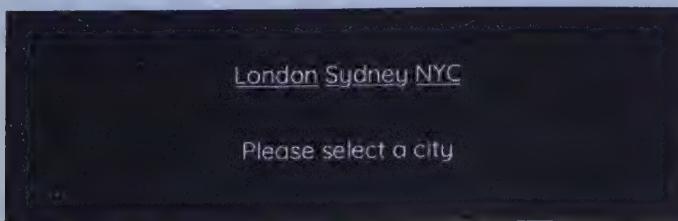
```
<?php
$cities = [
    'London' => '48 Store Street, WC1E 7BS',
    'Sydney' => '151 Oxford Street, 2021',
    'NYC'      => '1242 7th Street, 10492',
];
① $city  = $_GET['city'] ?? '';
② $valid = array_key_exists($city, $cities);

③ if ($valid) {
④     $address = $cities[$city];
⑤ } else {
⑥     $address = 'Please select a city';
}
?>

...
<?php foreach ($cities as $key => $value) { ?>
    <a href="get-3.php?city=<?= $key ?>"><?= $key ?></a>
<?php } ?>

<h1><?= $city ?></h1>
<p><?= $address ?></p>
```

ERGEBNIS



Dieses Beispiel baut auf den vorherigen auf und setzt auf Datenvalidierung, um zu überprüfen, ob der Query-String einen gültigen Ort enthält.

1. Wenn der Query-String eine Stadt enthält, wird diese in der Variablen `$city` gespeichert. Wenn nicht, wird `$city` eine leere Zeichenkette zugewiesen.
2. Die Funktion `array_key_exists()` überprüft, ob der Wert in `$city` ein Schlüssel des Arrays `$cities` ist. Trifft dies zu, wird die Variable `$valid` auf `true` gesetzt. Wenn nicht, erhält `$valid` den Wert `false`.
3. Die Variable `$valid` wird in der Bedingung einer `if`-Anweisung verwendet. Wenn der in ihr gespeicherte Wert `true` ist, wird der erste Code-Block ausgeführt.
4. Die Adresse für diese Stadt wird aus dem Array `$cities` abgerufen und in der Variablen `$address` gespeichert.
5. Ist der Wert von `$valid` `false`, wird der zweite Code-Block ausgeführt.
6. Die Variable `$address` enthält eine Nachricht, die den Benutzer zur Auswahl einer Stadt auffordert.

Probieren Sie es: Geben Sie in der Adressleiste des Browsers den Städtenamen Shanghai in den Query-String ein:

get-3.php?city=Shanghai

EINE FEHLERSEITE ANZEIGEN, WENN DIE DATEN NICHT KOMPLETT SIND

Wenn eine Seite Daten aus dem Query-String benötigt, diese aber fehlen oder ungültig sind, kann der PHP-Interpreter den Browser anweisen, eine andere Datei mit einer Fehlermeldung anzufordern.

Die Validierung der Daten im Query-String ist wichtig, denn wenn jemand auf Ihre Website verlinkt, passiert es leicht, dass versehentlich Daten aus dem Query-String verloren gehen.

Sie sollten von den Besucherinnen und Besuchern Ihrer Seite nicht erwarten, dass sie die Daten im Query-String bearbeiten können. Sollten die Daten also ungültig sein, können Sie weiterhelfen, indem Sie:

- Ihnen auf der Seite eine Meldung anzeigen. Diese könnte besagen, dass die angeforderte Seite nicht gefunden werden konnte, oder sie enthält die Aufforderung, aus einer Liste von Optionen auszuwählen (wie im Beispiel auf der vorherigen Seite).
- sie auf eine andere Seite schicken, die eine Fehlermeldung enthält.

Die nachstehende Bedingung überprüft, ob der in `$valid` gespeicherte Wert nicht `true` ist. Dementsprechend wären die Daten ungültig.

Auf Seite 226 haben Sie gesehen, dass Sie mit der in PHP eingebauten Funktion `header()` den Location-Header setzen können, den der PHP-Interpreter an den Browser sendet. Dies weist den Browser an, eine andere Seite anzufordern.

Wenn eine Seite wegen ungültiger Daten nicht dargestellt werden kann, sollten Sie den vom PHP-Interpreter an den Browser zurückgesendeten Statuscode aktualisieren (siehe S. 181). Dadurch verhindern Sie, dass Suchmaschinen falsche URLs in ihre Suchergebnisse aufnehmen. Die in PHP eingebaute Funktion `http_response_code()` dient zum Setzen des HTTP-Statuscodes. Als einziges Argument erwartet sie den zu übermittelnden Statuscode. Die Rückgabe des Statuscodes 404 bedeutet, dass die angeforderte Seite nicht gefunden werden konnte.

Nachdem der Statuscode und der Header gesetzt wurden, unterbricht der `exit`-Befehl die weitere Verarbeitung von Code auf der Seite (da hierbei ein Fehler auftreten könnte).

```
WENN NICHT VALIDE → if (!$valid) {  
    SETZE STATUSCODE → http_response_code(404);  
    AUF FEHLERSEITE UMLEITEN → header('Location: page-not-found.php');  
    CODE-AUSFÜHRUNG ABBRECHEN → exit;  
}
```

BESUCHER AUF EINE FEHLERSEITE UMLEITEN

PHP

```
section_b/c06/get-4.php

<?php
$cities = [
    'London' => '48 Store Street, WC1E 7BS',
    'Sydney' => '151 Oxford Street, 2021',
    'NYC'      => '1242 7th Street, 10492',
];
$city  = $_GET['city'] ?? '';
❶ $valid = array_key_exists($city, $cities);

❷ if (!$valid) {
❸     http_response_code(404);
❹     header('Location: page-not-found.php');
❺     exit;
}
$address = $cities[$city];
?>
...
<?php foreach ($cities as $key => $value) { ?>
    <a href="get-4.php?city=<?= $key ?>"><?= $key ?></a>
<?php } ?>

<h1><?= $city ?></h1>
<p><?= $address ?></p>
```

ERGEBNIS



Hinweis: Im Ergebniskasten wird die Datei `page-not-found.php` angezeigt, da der Query-String keinen Städtenamen enthält.

In diesem Beispiel werden Besucher auf eine Fehlerseite geleitet, wenn die Daten im Abfrage-String keine gültige Stadt enthalten.

1. Die PHP-Funktion `array_key_exists()` überprüft, ob der Städtename aus dem Query-String einer der Schlüssel des Arrays `$cities` ist. Die Funktion liefert in diesem Fall `true` zurück und ansonsten `false`. Dieser Wert wird in der Variablen `$valid` gespeichert.

2. Die Bedingung einer `if`-Anweisung überprüft, ob der in `$valid` gespeicherte Wert nicht `true` ist. (Der Operator `!` zeigt an, dass er nicht `true` sein sollte.) Lautet der Wert `false`, wird der nachfolgende Code-Block ausgeführt.

3. Die PHP-Funktion `http_response_code()` weist den PHP-Interpreter an, den Statuscode 404 an den Browser zurückzugeben. Dieser besagt, dass die Seite nicht gefunden werden konnte.

4. Die PHP-Funktion `header()` weist den PHP-Interpreter an, einen Location-Header anzufügen, der den Browser zur Anforderung der alternativen Datei `page-not-found.php` anweist.

5. Der PHP-Befehl `exit` weist den PHP-Interpreter an, keinen weiteren Code in der Datei auszuführen.

Wenn der Wert in `$valid true` ist, werden die Schritte 3–5 ignoriert, und die Seite wird angezeigt.

AUSGABE MASKIEREN

Wenn zuvor an den Server übermittelte Werte auf einer Seite angezeigt werden, müssen sie maskiert werden, damit Hacker sie nicht zur Ausführung bösartiger Skripte missbrauchen können.

Beim Maskieren von Daten werden Zeichen, die in einem Wert nicht vorkommen dürfen, entfernt (oder gegebenenfalls ersetzt). HTML besitzt beispielsweise fünf reservierte Zeichen, die von Browsern als Code behandelt werden:

< und > werden in Tags verwendet

" und ' enthalten Attributwerte

& dient zum Erzeugen von Entitäten

Um diese fünf Zeichen auf einer Seite darzustellen, müssen sie entweder durch einen Entitätsnamen oder eine Entitätsnummer als Platzhalter ersetzt werden. Browser zeigen dann die entsprechenden Zeichen an, statt sie als Code zu behandeln.

< > & " !

< > & " '
 ; <> & " ' ;

Wenn eine Seite Werte von einem Besucher empfängt und diese Werte dann wiederum auf einer Seite anzeigen muss, sollte eine Überprüfung auf diese fünf reservierten Zeichen erfolgen, die dann durch ihre entsprechenden HTML-Entitäten ersetzt werden. Dies gelingt mit der in PHP integrierten Funktion `htmlspecialchars()` (siehe Seite 246).

Wenn Sie die reservierten HTML-Zeichen nicht durch Entitäten ersetzen, können Hacker Werte übermitteln, die eine JavaScript-Datei mit Schadcode laden. Dies wird als Cross-Site-Scripting-Attacke (XSS) bezeichnet.

Wenn eine Besucherin zum Beispiel den folgenden Benutzernamen angibt und die Seite dann versucht, diesen anzuzeigen, könnte das Skript ausgeführt werden.

Luke<script src="http://eg.link/bad.js">
</script>

Werden die reservierten Zeichen durch Entitäten ersetzt, sehen die Besucher den obigen Text (und das Skript wird nicht ausgeführt). Im HTML-Quellcode für die Seite würde der Benutzername wie folgt aussehen:

Luke<script src="http://
eg.link/bad.js"></
script>

Von den Benutzern bereitgestellte Daten sollten nur im auf der Webseite sichtbaren HTML-Markup (oder in den Elementen `<title>` oder `<meta>`) erscheinen. Zeigen Sie keine von Benutzern gelieferten Daten in:

- Kommentaren in Ihrem Code
- CSS-Regeln (da sie ein Skript in eine Seite einbinden können)
- `<script>`-Elementen
- Tag-Namen
- Attributnamen
- als Wert von HTML-Event-Attributen wie `onclick` und `onload`
- als Wert eines HTML-Attributs, das Dateien lädt (wie etwa das `src`-Attribut)

Wie Sie auf Seite 280 erfahren, müssen auch Werte, die in einer URL oder einem Query-String verwendet werden, maskiert werden.

RISIKO DER UNMASKIERTEN AUSGABE

PHP

```
section_b/c06/xss-1.php  
① <a class="badlink" href="xss_1.php?msg=<script>  
    src=js/bad.js</script>">LINK TO DEMONSTRATE XSS</a>  
  
② <?php  
$message = $_GET['msg'] ?? 'Click link at top of page';  
?>  
...  
③ <h1>XSS Example</h1>  
<p><?= $message ?></p>
```

ERGEBNIS



Dieses Beispiel zeigt, was passiert, wenn die Daten nicht maskiert wurden.

1. Für dieses Beispiel zeigen wir einen Link auf dieselbe Seite. Der Link enthält einen Query-String mit `<script>`-Tags. (Bei einer echten XSS-Attacke könnte der Link auf diese Seite auf einer anderen Website, in einer E-Mail oder einer anderen Art von Nachricht erscheinen).

2. Die PHP-Seite überprüft das superglobale Array `$_GET`, um festzustellen, ob der Query-String den Namen `msg` enthält.

- Trifft dies zu, wird der entsprechende Wert in der Variablen `$message` gespeichert.

- Ist dies nicht der Fall, wird in `$message` eine Aufforderung zum Anklicken des Links abgelegt.

3. Der Wert in `$message` wird auf der Seite angezeigt.

Wenn Sie auf den Link am oberen Seitenrand klicken, wird das Skript ausgeführt, da der Wert im Query-String nicht maskiert wurde.

Hinweis: Wie Sie auf der nächsten Seite sehen werden, bewirkt das Maskieren von Text im Query-String, dass die Skript-Tags auf der Seite angezeigt werden und der Browser sie nicht als Code behandelt.

RESERVIERTE HTML-ZEICHEN MASKIEREN

Die in PHP integrierte Funktion `htmlspecialchars()` ersetzt die reservierten Zeichen in HTML durch ihre jeweiligen Entitäten, sodass diese Zeichen angezeigt werden und nicht mehr als Code ausgeführt werden können.

Die Funktion `htmlspecialchars()` hat vier Parameter; der erste ist erforderlich, alle weiteren sind optional.

- `$text` ist der zu maskierende Text.
- `$flag` ist eine Option, die bestimmt, welche Zeichen kodiert werden (die untere Tabelle zeigt die üblichen Optionen).
- `$encoding` ist das Kodierungsverfahren für den String (der Standardwert lautet UTF-8, wenn nichts angegeben wird).
- `$double_encode` HTML-Entitäten beginnen mit einem kaufmännischen Und. In Zeichenketten, die eine Entität enthalten, wird daher standardmäßig das kaufmännische Und kodiert, und die Seite zeigt die Entität an (und nicht das reservierte Zeichen). Der Wert `false` für diesen Parameter weist den PHP-Interpreter an, Entitäten in der Zeichenkette nicht zu kodieren.

```
htmlspecialchars($text[, $flag][, $encoding][, $double_encode]);
```

FLAG	BESCHREIBUNG
ENT_COMPAT	Doppelte Anführungszeichen umwandeln, einfache Anführungszeichen belassen (Standardeinstellung, wenn kein Flag angegeben wird)
ENT_QUOTES	Doppelte und einfache Anführungszeichen umwandeln
ENT_NOQUOTES	Doppelte und einfache Anführungszeichen nicht umwandeln
ENT_SUBSTITUTE	Damit die Funktion keinen leeren String zurückgibt, ungültige Zeichen durch Ersatzzeichen ersetzen (in UTF-8 ist das U+FFFD, in jeder anderen Kodierung ist es �)
ENT_HTML401	Code als HTML 4.01 behandeln
ENT_HTML5	Code als HTML 5 behandeln
ENT XHTML	Code als XHTML behandeln

Um mehrere Flags anzugeben, trennen Sie diese durch ein Pipe-Symbol so wie in `ENT_QUOTES|ENT_HTML5`.

von Nutzern übermittelte Inhalte maskieren

PHP

section_b/c06/xss-2.php

```
<a class="badlink" href="xss-2.php?msg=<script>
src=js/bad.js</script>">ESCAPING MARKUP</a>

<?php
$message = $_GET['msg'] ?? 'Click the link above';
?> ...
<h1>XSS Example</h1>
① <p><?= htmlspecialchars($message) ?></p>
```

PHP

section_b/c06/xss-3.php

```
<a class="badlink" href="xss-3.php?msg=<script>
src=js/bad.js</script>">ESCAPING MARKUP</a>

<?php
function html_escape(string $string): string
{
    return htmlspecialchars($string,
        ENT_QUOTES|ENT_HTML5, 'UTF-8', true);
}
$message = $_GET['msg'] ?? 'Click the link above';
?> ...
<h1>XSS Example</h1>
③ <p><?= html_escape($message) ?></p>
```

ERGEBNIS

XSS EXAMPLE

<script src=js/bad.js></script>

1. Das erste Beispiel weist nur eine Änderung gegenüber dem vorherigen Beispiel auf: Wenn der Wert in \$message ausgegeben wird, dient die PHP-Funktion htmlspecialchars() dazu, die reservierten HTML-Zeichen durch die jeweiligen Entitäten zu ersetzen. Wird also auf den Link geklickt, zeigt der Browser den HTML-Code für die <script>-Tags auf dem Bildschirm an, anstatt ihn auszuführen.

2. Eine zweite Version des gleichen Beispiels wurde um die benutzerdefinierte Funktion html_escape() ergänzt. Sie nimmt eine Zeichenkette als Argument entgegen und gibt diese wieder zurück, wobei alle reservierten Zeichen durch HTML-Entitäten ersetzt werden. Beim Aufruf von htmlspecialchars() werden Werte für alle vier Parameter übergeben.

3. Die Funktion html_escape() wird aufgerufen, um die Nachricht aus dem Query-String auszugeben. Das Ergebnis der beiden Beispiele sieht genau gleich aus.

Hinweis: Im Code-Download für dieses Kapitel finden Sie die Definition der Funktion html_escape() auch in der Include-Datei functions.php.

Probieren Sie es: Ersetzen Sie in Schritt 2 die Funktionsdefinition durch eine include-Anweisung zum Einbinden der Datei functions.php.

WIE FORMULARDATEN AN DEN SERVER GESENDET WERDEN

Formulare ermöglichen es Besuchern, Text einzugeben und Optionen auszuwählen. Für jedes Formularelement kann der Browser einen Namen und einen Wert zusammen mit einer Seitenanfrage an den Server senden.

Der HTML-Tag <form> erfordert zwei Attribute:

- Der Wert des action-Attributs ist die PHP-Datei, an die die Formulardaten gesendet werden sollen.
- Der Wert des method-Attributs legt fest, wie die Formulardaten an den Server gesendet werden sollen.

Das method-Attribut sollte einer dieser beiden Werte haben:

- GET versendet die Formulardaten über HTTP GET in einem Query-String, der hinten an die URL angehängt wird.
- POST versendet die Daten über HTTP POST in den HTTP-Headern, die der Browser an den Server schickt.

EMPFÄNGERSEITE FÜR DIE DATEN

```
<form action="join.php" method="POST">
    <p>Email: <input type="email" name="email"></p>
    <p>Age: <input type="number" name="age"></p>
    <p><input type="checkbox" name="terms" value="true">
        I agree to the terms and conditions.</p>
    <input type="submit" value="Save">
</form>
```

HTTP-METHODE ZUM SENDEN DER DATEN

Wenn der Benutzer das Formular abschickt, fordert der Browser die im action-Attribut angegebene Seite an.

Der Wert des action-Attributs kann ein relativer Pfad von der Seite, die das Formular erstellt, zu der Seite, die das Formular verarbeitet, sein oder eine vollständige URL.

Häufig wird das Formular an dieselbe PHP-Seite übermittelt, die auch zur Anzeige des Formulars verwendet wurde.

Das obige Formular wird mittels HTTP POST gesendet, sodass der Browser die Namen und Werte der Formularelemente an die HTTP-Header anfügt. Die Header werden mit der Anforderung von join.php übermittelt. Für jeden Header ist:

- der Name der Wert des name-Attributs des betreffenden Formularelements.
- der Wert der vom Benutzer eingegebene Text oder der Wert des von ihm ausgewählten Elements.

Die nachstehenden HTML-Formularelemente lassen sich in zwei Kategorien einteilen: Eingabefelder für Text und Auswahlmöglichkeiten für eine bestimmte Option.

Wenn eine Besucherin eine Texteingabe vornimmt, wird als Name der Wert des Attributs name an den Server gesendet und als Wert der eingegebene Text. Gibt die Anwenderin keinen Text für dieses Formularelement ein, wird der Name trotzdem an den Server übermittelt, und der Wert entspricht einer leeren Zeichenkette.

Wurde eine Option ausgewählt, entspricht der Name dem Wert des Attributs name und der Wert den Daten im value -Attribut für die ausgewählte Option. Hat die Benutzerin keine Option ausgewählt, sendet der Browser für dieses Formularelement keinerlei Daten an den Server.

EINGABEFELD	BEISPIEL	ZWECK
Texteingabe	<input type="text" name="username">	einzelne Textzeile eingeben
Zahleneingabe	<input type="number" name="age">	Zahl eingeben
E-Mail-Eingabe	<input type="email" name="email">	E-Mail-Adresse eingeben
Passwort	<input type="password" name="password">	Passwort eingeben
Textbereich	<textarea name="bio"></textarea>	Längerer Text eingeben

AUSWAHLMÖGLICHKEIT	BEISPIEL	ZWECK
Optionsfelder	<input type="radio" name="rating" value="good"> <input type="radio" name="rating" value="bad">	Eine von mehreren Optionen wählen
Auswahlfelder	<select name="preferences"> <option value="email">Email</option> <option value="phone">Phone</option> </select>	Aus mehreren Optionen wählen
Kontrollfelder	<input type="checkbox" name="terms" value="true">	Eine einzelne Option wählen

Um die serverseitige Validierung zu demonstrieren, beschränkt sich dieses Buch auf die Validierung von Daten auf dem Server. Reale Websites sollten JavaScript sowie Zahlen- und E-Mail-Eingabefelder verwenden, um Daten bereits vor der Übertragung an den Server im Browser zu validieren. Auf dem Server werden die Daten dann erneut validiert (da sich die Überprüfung im Browser umgehen lässt).

Hinweis: Wenn der PHP-Interpreter Daten vom Browser in ein superglobales Array übernimmt, sind diese immer vom Datentyp String, auch wenn der Wert eine Zahl oder ein boolescher Wert ist.

Im nächsten Kapitel erfahren Sie mehr über Datei-uploads zu Servern und deren Überprüfung.

FORMULARDATEN ABRUFEN

Wenn der PHP-Interpreter Daten empfängt, die über HTTP POST gesendet wurden, fügt er diese dem superglobalen Array `$_POST` hinzu.

Wenn der Besucher ein Formular mittels HTTP POST absendet, empfängt der PHP-Interpreter die Seitenanforderung und fügt die Formulardaten (aus den übermittelten HTTP-Headern) an das superglobale Array `$_POST` an. Dabei gilt:

- Der **Schlüssel** ist der Name des Formularelements.
- Der **Wert** ist der Wert, den der Benutzer eingegeben oder ausgewählt hat.

Der Code in der PHP-Datei kann auf die Werte im superglobalen Array `$_POST` genauso zugreifen wie auf die Werte eines beliebigen anderen assoziativen Arrays. Handelt es sich bei dem Formularelement um ein Texteingabefeld, wird dieses immer einen Wert enthalten (sofern es nicht deaktiviert wurde):

```
$email = $_POST['email'];
```

VARIABLE

SCHLÜSSEL

Handelt es sich bei dem Formularelement um eine Auswahlmöglichkeit, werden der Name und der Wert nur dann in die HTTP-Header übernommen, wenn der Besucher eine Auswahl trifft. Deshalb werden mit dem Null-Koaleszenzoperator Informationen aus dem superglobalen Array `$_POST` erfasst (genauso wie er zur Werteerfassung aus dem Query-String verwendet wurde).

```
$age = $_POST['age'] ?? false;
```

VARIABLE

SCHLÜSSEL

STANDARD-

WERT

Wenn das Formular mittels HTTP GET übertragen wurde, entnimmt der PHP-Interpreter die Formulardaten dem Query-String und fügt sie an das superglobale Array `$_GET` an.

Das Beispiel auf der rechten Seite zeigt den Inhalt der superglobalen Arrays für Seiten, die Formulare verwenden. Der superglobale Array wird mit der Funktion `var_dump()` (siehe Seite 192) angezeigt. So können Sie sehen, welche Elemente dem Array hinzugefügt wurden, und auch, dass alle Daten in diesen superglobalen Arrays vom Datentyp String sind – selbst wenn es sich um eine Zahl oder einen booleschen Wert handelt.

Es ist wichtig, dieses Beispiel selbst auszuprobieren und zu sehen, wie sich die Daten im superglobalen Array ändern, wenn:

- die Seite bereits geladen wird, bevor das Formular abgeschickt wurde.
- das Formular leer und ohne Dateneingabe abgeschickt wird.
- die Formularfelder ausgefüllt werden.

FORMULARDATEN ENTGEGENNEHMEN

PHP

section_b/c06/collecting-form-data.php

```
<form action="collecting-form-data.php" method="POST">
    <p>Name: <input type="text" name="name"></p>
    <p>Age: <input type="text" name="age"></p>
    <p>Email: <input type="text" name="email"></p>
    <p>Password: <input type="password" name="pwd"></p>
    <p>Bio: <textarea name="bio"></textarea></p>
    <p>Contact preference:<br/>
        <select name="preferences">
            <option value="email">Email</option>
            <option value="phone">Phone</option>
        </select></p>
    <p>Rating:<br/>
        1 <input type="radio" name="rating" value="1">&ampnbsp
        2 <input type="radio" name="rating" value="2">&ampnbsp
        3 <input type="radio" name="rating" value="3"></p>
    <p><input type="checkbox" name="terms" value="true">
        I agree to the terms and conditions.</p>
    <p><input type="submit" value="Save"></p>
</form>
③ <pre><?php var_dump($_POST); ?></pre>
```

ERGEBNIS



1. Fünf Textfelder fragen nach Namen, Alter, E-Mail-Adresse, Passwort und dem Lebenslauf des Benutzers.

2. Drei Formularelemente bieten dem Benutzer verschiedene Auswahlmöglichkeiten an.

3. Der Inhalt des superglobalen Arrays `$_POST` wird mit der Funktion `var_dump()` ausgegeben.

Bei Laden der Seite wurde das Formular noch nicht abgeschickt, daher ist das superglobale Array `$_POST` leer.

Wird das Formular leer abgeschickt, enthält das superglobale Array `$_POST` ein Element für jedes Textfeld; der Wert ist jeweils eine leere Zeichenkette. Das Auswahlfeld wird mit dem Standardwert, der beim Laden der Seite angezeigt wird, an den Server gesendet. Die Namen und Werte der Optionsfelder und des Kontrollkästchens werden jedoch nicht an den Server übermittelt.

Wenn alle Formularelemente ausgefüllt sind, enthält das superglobale Array `$_POST` ein Element für jedes Formularelement. Alle an den Server gesendeten Werte sind Strings.

Probieren Sie es: Ändern Sie den Wert im `method`-Attribut des `<form>`-Tags auf `GET`, um die Daten über HTTP GET zu senden. Lassen Sie dann in Schritt 3 den Inhalt des superglobalen Arrays `$_GET` anzeigen.

SO PRÜFEN SIE, OB EIN FORMULAR ÜBERMITTELT WURDE

Bevor Sie die Daten eines Formulars erfassen und verarbeiten können, muss es natürlich zuerst abgeschickt werden. Je nach Übertragungsmethode (HTTP POST oder HTTP GET) gibt es unterschiedliche Techniken, um zu prüfen, ob das Formular abgeschickt wurde.

HTTP POST

Das superglobale Array `$_SERVER` (siehe Seite 190) speichert im Schlüssel `REQUEST_METHOD` die HTTP-Methode, mit der die Seite angefordert wurde. Wenn ein Formular mit HTTP POST übermittelt wird, hat er den Wert `POST`.

Um zu erfahren, ob ein Formular per HTTP POST übermittelt wurde, überprüft die Bedingung einer `if`-Anweisung, ob der Schlüssel `REQUEST_METHOD` dem Wert `POST` entspricht. Der Code zur Verarbeitung des Formulars steht im nachfolgenden Code-Block.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    // Code zum Abrufen und Verarbeiten der Formulardaten hierher  
}
```

HTTP GET

Wenn ein Benutzer einen Link anklickt oder eine URL in die Adressleiste des Browsers eingibt, wird die Anfrage immer über HTTP GET gesendet. Daher können Sie das superglobale Array `$_SERVER` nicht verwenden, um zu prüfen, ob ein Formular über HTTP GET versendet wurde. Stattdessen können Sie eines von Beidem hinzufügen:

- verstecktes Eingabefeld im Formular oder
- Name und Wert für die Absenden-Schaltfläche

Wenn das Formular abgeschickt wird, werden der Name und der Wert des versteckten Eingabefelds oder der send-Schaltfläche zum superglobalen Array `$_GET` hinzugefügt.

Die Bedingung einer `if`-Anweisung kann dann kontrollieren, ob das superglobale Array `$_GET` den beim Absenden des Formulars übermittelten Wert enthält. Trifft dies zu, kann der Code zur Erfassung und Verarbeitung der Daten ausgeführt werden.

```
$submitted = $_GET['submitted'] ?? '';  
if ($submitted == 'true') {  
    // Code zum Abrufen und Verarbeiten der Formulardaten hierher  
}
```

ÜBERPRÜFEN, OB EIN FORMULAR ABGESCHICKT WURDE

PHP

section_b/c06/check-for-http-post.php

```
<?php  
① if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
②     $term = $_POST['term'];  
    echo 'You searched for ' . htmlspecialchars($term);  
} else { ?>  
    <form action="check-for-http-post.php" method="POST">  
        Search for: <input type="text" name="term">  
        <input type="submit" value="search">  
    </form>  
<?php } ?>
```

PHP

section_b/c06/check-for-http-get.php

```
<?php  
④ $submitted = $_GET['sent'] ?? '';  
if ($submitted === 'search') {  
    $term = $_GET['term'] ?? '';  
    echo 'You searched for ' . htmlspecialchars($term);  
} else { ?>  
    <form action="check-for-http-get.php" method="GET">  
        Search for: <input type="search" name="term">  
        <input type="submit" name="sent" value="search">  
    </form>  
<?php } ?>
```

ERGEBNIS



1. Die Bedingung einer `if`-Anweisung untersucht das superglobale Array `$_SERVER` daraufhin, ob der Schlüssel `REQUEST_METHOD` den Wert `POST` hat.

2. Ist dies der Fall, wurde das Suchformular per HTTP POST gesendet, und eine Meldung zur Anzeige des Suchbegriffs wird eingesetzt.

3. Andernfalls wird dies übersprungen und das Formular angezeigt.

In diesem Beispiel wird der Name der Absenden-Schaltfläche übermittelt, und ihr Wert ist `search`. Wurde das Formular versendet, werden beide dem superglobalen Array `$_GET` hinzugefügt.

4. Der Null-Koaleszenz-Operator prüft, ob das superglobale Array `$_GET` einen Wert für den gesendeten Schlüssel enthält. Trifft dies zu, wird der Wert in der Variablen `$submitted` gespeichert; andernfalls wird sie mit einer leeren Zeichenkette gefüllt.

5. Die Bedingung einer `if`-Anweisung überprüft, ob der Wert in `$submitted` `search` entspricht. Trifft dies zu, wurde das Formular über HTTP GET gesendet, und der Suchbegriff wird angezeigt.

6. Andernfalls wird das Formular angezeigt.

Probieren Sie es: Verwenden Sie ein verstecktes Eingabefeld, um zu signalisieren, dass das Formular abgeschickt wurde.

ZAHLEN VALIDIEREN

Erfasste Formulardaten sollten validiert werden, um sicherzustellen, dass alle erforderlichen Werte eingegeben wurden und die Daten im richtigen Format vorliegen. Dies verhindert Fehler bei der Seitenausführung.

Um zu prüfen, ob ein Wert eine Zahl ist, verwenden Sie die PHP-eigene Funktion `is_numeric()` (Seite 216). Müssen Sie überprüfen, ob die Zahl in einem bestimmten Bereich aus erlaubten Zahlen liegt, können Sie hierfür eine eigene Funktion definieren. Unten sehen Sie eine Funktion, die mithilfe von Vergleichsoperatoren prüft, ob eine Zahl innerhalb des zulässigen Wertebereichs liegt (zwischen Mindest- und Höchstwert). Die Funktion hat drei Parameter:

- `$number` ist der zu überprüfende Wert
- `$min` ist der zulässige Mindestwert
- `$max` ist der zulässige Höchstwert

```
function is_number($number, int $min = 0, int $max = 100): bool
{
    return ($number >= $min and $number <= $max);
}
```

IST DER WERT >= MINIMUM?

IST DER WERT <= MAXIMUM?

Sind die Formulardaten ungültig, wird das Formular oft noch einmal angezeigt, damit die betreffende Person es erneut versuchen kann. In solchen Fällen kann die eingegebene Zahl im Formularelement angezeigt werden, indem sie in das `value`-Attribut des `<input>`-Tags geschrieben wird. Die Funktion `htmlspecialchars()` dient bei der Darstellung des Werts dazu, einen XSS-Angriff zu verhindern.

Die Bedingung in der Funktion enthält zwei Ausdrücke, die prüfen, ob die Zahl:

- größer oder gleich dem Mindestwert ist
- kleiner oder gleich dem Höchstwert ist

Wenn beide Ausdrücke den Wert `true` ergeben, liefert die Funktion ebenfalls `true` zurück. Wenn einer der Ausdrücke `false` liefert, gibt die Funktion `false` zurück.

Wenn eine Seite eine Zahl erfasst hat, kann sie durch Aufruf dieser Funktion prüfen, ob der Wert gültig ist.

Da der vom Anwender eingegebene Wert nur beim Absenden des Formulars erfasst wird, muss die Variable `$age` am Seitenanfang deklariert werden und als Anfangswert einen leeren String erhalten. Wenn die Variable nicht am Anfang der Seite deklariert wäre, würde der Versuch, sie im `value`-Attribut des Formularsteuerelements anzuzeigen, zu einem `Undefined variable` error im Eingabefeld führen.

```
<input type="text" name="age" value="<?= htmlspecialchars($age) ?>">
```

EINE ZAHL AUF GÜLTIGKEIT ÜBERPRÜFEN

PHP

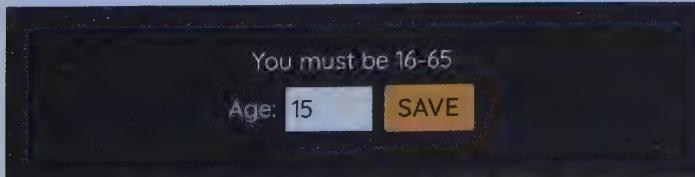
section_b/c06/validate-number-range.php

```
<?php
declare(strict_types = 1);
① $age      = '';
② $message = '';

function is_number($number, int $min = 0, int $max = 100): bool
{
    return ($number >= $min and $number <= $max);

③ if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    ④ $age      = $_POST['age'];
    ⑤ $valid = is_number($age, 16, 65);
    ⑥ if ($valid) {
        $message = 'Age is valid';
    } else {
        $message = 'You must be 16-65';
    }
}
?> ...
⑧ <?= $message ?>
<form action="validate-number-range.php" method="POST">
    Age: <input type="text" name="age" size="4"
            value=<?= htmlspecialchars($age) ?>>
    <input type="submit" value="Save">
</form>
```

ERGEBNIS



1. Die Variablen \$age und \$message werden initialisiert und mit leeren Zeichenketten belegt.
2. Die Funktion is_number() (siehe linke Seite) wird definiert.
3. Die Seite prüft, ob das Formular abgeschickt wurde. Wenn ja ...
4. Das Alter wird aus dem superglobalen Array \$_POST ausgelesen. Die Daten stammen aus einem Eingabefeld, sodass beim Abschicken des Formulars stets ein Wert dafür gesendet wird.
5. Die Funktion is_number() wird aufgerufen. Der vom Benutzer eingegebene Wert dient als erstes Argument, und die Zahlen 16 und 65 sind der erforderliche Mindest- und Höchstwert. Der zurückgelieferte boolesche Wert wird in \$valid gespeichert.
6. Die Bedingung einer if-Anweisung überprüft, ob der Wert in \$valid true ist. Wenn ja, wird in der Variablen \$message eine Nachricht gespeichert, die besagt, dass die Altersprüfung bestanden wurde.
7. Andernfalls wird in \$message eine Fehlermeldung abgelegt.
8. Die Nachricht wird angezeigt.
9. Die vom Benutzer eingegebene Zahl (oder der Anfangswert aus Schritt 1) wird mit htmlspecialchars() im Zahleneingabefeld angezeigt.

TEXTLÄNGE ÜBERPRÜFEN

Websites beschränken oft die maximal darstellbare Zeichenanzahl etwa in Benutzernamen, Beiträgen, Artikeltiteln und Profilen. Mit einer einzigen Funktion lässt sich die Länge jeder empfangenen Zeichenkette ermitteln.

So überprüfen Sie, ob eine Texteingabe zwischen einer Mindest- und einer Höchstanzahl von Zeichen liegt:

- Die in PHP eingebaute Funktion `mb_strlen()` (siehe Seite 210) zählt, wie viele Zeichen die Zeichenkette enthält. Dieser Wert wird in einer Variablen gespeichert.
- Dann überprüft eine Bedingung mithilfe von zwei Ausdrücken, ob die Zeichenanzahl sich innerhalb des zulässigen Bereichs bewegt (auf dieselbe Weise, wie sie auf der vorhergehenden Seite eingesetzt wurden, um zu überprüfen, ob eine Zahl innerhalb eines zulässigen Bereichs liegt).

Wenn die Zeichenanzahl gültig ist, gibt die Funktion `true` zurück, wenn nicht, dann `false`.

Wenn Code zur Datenvalidierung in einer Funktion platziert wird, lässt er sich zur Überprüfung mehrerer Formularelemente nutzen. Damit sparen Sie sich die Wiederholung von Code für dieselbe Aufgabe.

Die folgende Funktion (und das vorherige Beispiel) nutzen Parameter, sodass die Mindest- und Höchstwerte jedem Aufruf unterschiedlich sein können.

Wenn mehrere Seiten dieselben Validierungsaufgaben durchführen, sollten Sie die Funktionsdefinitionen in einer Include-Datei zusammenfassen. Dann können Sie diese Datei einbinden, anstatt die gleichen Funktionsdefinitionen auf jeder Seite zu wiederholen. Im Code-Download für dieses Kapitel finden Sie die Include-Datei `validate.php`, die drei der hier besprochenen Funktionsdefinitionen enthält.

```
function is_text($text, int $min = 0, int $max = 100): bool
{
    $length = mb_strlen($text);
    return ($length >= $min and $length <= $max);
}
```

IST DER WERT >= MINIMUM?

IST DER WERT <= MAXIMUM?

LÄNGE EINES STRINGS KONTROLIEREN

PHP

section_b/c06/validate-text-length.php

```
<?php
declare(strict_types = 1);
① $username = '';
② $message = '';

function is_text($text, int $min = 0, int $max = 1000): bool
{
    $length = mb_strlen($text);
    return ($length >= $min and $length <= $max);
}

③ if ($_SERVER['REQUEST_METHOD'] == 'POST') {
④     $username = $_POST['username'];
⑤     $valid    = is_text($username, 3, 18);
⑥     if ($valid) {
        $message = 'Username is valid';
    } else {
        $message = 'Username must be 3-18 characters';
    }
}
?> ...
⑧ <?= $message ?>
<form action="validate-text-length.php" method="POST">
    Username: <input type="text" name="username"
        value=<?= htmlspecialchars($username) ?>">
    <input type="submit" value="Save">
</form>
```

ERGEBNIS



1. Die Variablen \$username und \$message werden initialisiert.
2. Die benutzerdefinierte Funktion `is_text()` (auf der linken Seite vorgestellt) wird definiert.
3. Die Seite prüft, ob das Formular abgeschickt wurde. Wenn ja ...
4. Der Text wird aus dem superglobalen Array `$_POST` abgerufen.
5. Die Funktion `is_text()` wird aufgerufen, um zu prüfen, ob der eingegebene Text zwischen 3 und 18 Zeichen lang ist. Ihr Rückgabewert wird in `$valid` gespeichert.
6. Die Bedingung einer `if`-Anweisung überprüft, ob der Wert in `$valid` `true` ist. Ist dies der Fall, wird in `$message` eine Meldung geschrieben, dass der Benutzername gültig ist.
7. Andernfalls wird in `$message` eine Nachricht hinterlegt, die dem Benutzer mitteilt, dass der Benutzername zwischen 3 und 18 Zeichen lang sein muss.
8. Der Wert in der `$message`-Variable wird ausgegeben.
9. Der Wert in `$username` wird im Texteingabefeld angezeigt. Dies ist entweder der vom Anwender übermittelte Wert oder die leere Zeichenkette, mit der die Variable in Schritt 1 initialisiert wurde.

DATENVALIDIERUNG MIT REGULÄREN AUSDRÜCKEN

Mithilfe eines regulären Ausdrucks kann überprüft werden, ob die Benutzereingabe mit einem bestimmten Zeichenmuster übereinstimmt.

Wie Sie auf den Seiten 214 bis 217 gesehen haben, lassen sich mit regulären Ausdrücken zulässige Zeichenmuster beschreiben, etwa für Kreditkartennummern, Postleitzahlen und Telefonnummern. Die folgende Funktion nutzt reguläre Ausdrücke, um die Passwortstärke zu überprüfen.

Die Funktion enthält eine Bedingung mit vier Ausdrücken:

Zunächst kontrolliert `mb_strlen()`, ob der Wert mindestens 8 Zeichen enthält.

Anschließend wird die PHP-Funktion `preg_match()` dreimal aufgerufen, um zu überprüfen, ob das in einem regulären Ausdruck beschriebene Zeichenmuster im Passwort vorkommt.

Liefern alle Ausdrücke den Wert `true`, gibt der nachfolgende Code-Block den Wert `true` zurück (weil das Passwort die Anforderungen erfüllt). Andernfalls liefert er `false` zurück, falls die Funktion noch ausgeführt wird.

Als Parameter akzeptiert sie ein Passwort. Zuerst wird geprüft, ob es aus mindestens 8 Zeichen besteht. Mit regulären Ausdrücken wird dann geprüft, ob folgende Zeichen enthalten sind:

- Großbuchstaben
- Kleinbuchstaben
- Ziffern

Die einzelnen Checks werden mit dem Operator `and` verknüpft. Wenn alle Bedingungen erfüllt sind, gibt die Funktion den Wert `true` zurück, andernfalls den Wert `false`. (Die einzelnen Überprüfungen ließen sich auch mit einem einzigen regulären Ausdruck durchführen, der dann aber schwerer lesbar wäre).

```
function is_password(string $password): bool
{
    if (
        mb_strlen($password) >= 8
        and preg_match('/[A-Z]/', $password)
        and preg_match('/[a-z]/', $password)
        and preg_match('/[0-9]/', $password)
    ) {
        return true; // Alle Tests bestanden
    }
    return false; // Ungültig
}
```

Hinweis: Obwohl Browser Kennwörter bei der Eingabe maskieren, werden die Daten trotzdem als Klartext in den HTTP-Header übermittelt. Daher sollten personenbezogene Daten ausschließlich über HTTPS gesendet werden (siehe die Seiten 184 bis 185).

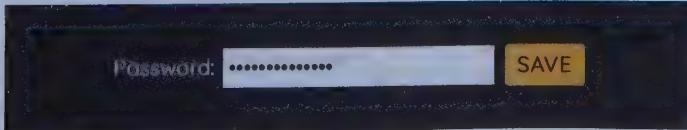
PASWORTSTÄRKE ÜBERPRÜFEN

PHP

section_b/c06/validate-password.php

```
<?php
declare(strict_types = 1);
① $password = '';
② $message = '';
③ function is_password(string $password): bool
{
    if (
        mb_strlen($password) >= 8
        and preg_match('/[A-Z]/', $password)
        and preg_match('/[a-z]/', $password)
        and preg_match('/[0-9]/', $password)
    ) {
        return true; // Alle Tests bestanden
    }
    return false; // Ungültig
}
⑥ if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    ⑦ $password = $_POST['password'];
    ⑧ $valid = is_password($password);
    ⑨ $message = $valid ? 'Password is valid' :
        'Password not strong enough';
}
?> ...
⑩ <?= $message ?>
<form action="validate-password.php" method="POST">
    Password: <input type="password" name="password">
    <input type="submit" value="Save">
</form>
```

ERGEBNIS



1. Die Variablen \$password und \$message werden initialisiert.

2. Die Funktion is_password() wird mit einem Parameter definiert: dem zu überprüfenden Passwort.

3. Eine if-Anweisung enthält vier Ausdrücke, von denen jeder true oder false ergibt. Sie sind durch den Operator and miteinander verknüpft, sodass der nachfolgende Code-Block nur ausgeführt wird, wenn alle Ausdrücke true ergeben.

4. Der Code-Block liefert true, und die Funktion wird nicht mehr weiter ausgeführt.

5. Andernfalls, wenn eine der Bedingungen nicht erfüllt wurde, liefert die Funktion false zurück.

6. Wenn das Formular abgeschickt wurde, wird der nachfolgende Code-Block ausgeführt.

7. Das Kennwort wird aus dem superglobalen Array \$_POST übernommen.

8. is_password() wird aufgerufen, um das Benutzerkennwort zu überprüfen. Das Ergebnis wird in der Variablen \$valid gespeichert.

9. Ein ternärer Operator überprüft, ob die Variable \$valid den Wert true hat. Wenn ja, wird in \$message eine Erfolgsmeldung eingetragen und andernfalls eine Fehlermeldung.

10. Der Wert in der Variablen \$message wird ausgegeben.

AUSWAHL- UND OPTIONSFELDER

Auswahl- und Optionsfelder bieten dem Benutzer die Möglichkeit, aus einer Liste von Auswahlmöglichkeiten auszuwählen. Der Browser sendet nur dann den Namen und den Wert an den Server, wenn eine Option ausgewählt wurde. Der Wert wird validiert, indem geprüft wird, ob er mit einer der Optionen übereinstimmt.

Wenn ein Formular ein Auswahl- oder Optionsfeld nutzt, können Sie ein indiziertes Array mit allen verfügbaren Auswahlmöglichkeiten erstellen und es in einer Variablen speichern.

Das folgende Array speichert Bewertungen von 1 bis 5 Sternen.

```
$star_ratings = [1, 2, 3, 4, 5,];
```

Um zu überprüfen, ob der Benutzer eine gültige Option ausgewählt hat, wird die in PHP integrierte Funktion `in_array()` verwendet.

Das Array kann dann verwendet werden, um:

- die Optionen in Auswahl- oder Optionsfeldern anzulegen.
- zu überprüfen, ob der Benutzer eine dieser Optionen ausgewählt hat.

```
$valid = in_array($stars, $star_ratings);
```

ÜBERMITTELTER
WERT

GÜLTIGE OPTIONEN

Zur Erstellung der Formularelemente können Sie die Auswahlmöglichkeiten in einer Schleife durchlaufen und für jede Option ein neues Element anlegen. Wird das Formular dem Benutzer erneut angezeigt, lässt sich die ausgewählte Option mithilfe eines ternären Operators hervorheben.

```
<?php foreach ($option as $star_ratings) { ?>
  <?= $option ?>
  <input type="radio" name="stars" value="<?= $option ?>">
  <?= ($stars == $option) ? 'checked' : '' ?>
<?php } ?>
```

Die Bedingung eines ternären Operators überprüft, ob der Wert in der Variablen `$stars` mit dem aktuellen Wert in der Schleife übereinstimmt. Wenn ja, wird das überprüfte Attribut hinzugefügt. Wenn nicht, wird stattdessen eine leere Zeichenkette ausgegeben.

Hinweis: Dieses Beispiel setzt voraus, dass die Variable `$stars` initialisiert wurde (siehe Schritt 1 auf der rechten Seite).

AUSWAHLMÖGLICHKEITEN VALIDIEREN

PHP

section_b/c06/validate-options.php

```
<?php  
① $stars  = '';  
② $message = '';  
③ $star_ratings = [1, 2, 3, 4, 5,];  
  
④ if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
⑤     $stars  = $_POST['stars'] ?? '';  
⑥     $valid   = in_array($stars, $star_ratings);  
⑦     $message = $valid ? 'Thank you' : 'Select an option';  
⑧ }  
⑨ ?> ...  
⑩ <?= $message ?>  
    <form action="validate-options.php" method="POST">  
        Star rating:  
        <?php foreach ($star_ratings as $option) { ?>  
        <?= $option ?> <input type="radio" name="stars"  
            value="<?= $option ?>"  
            <?= ($stars == $option) ? 'checked' : '' ?>>  
        <?php } ?>  
        <input type="submit" value="Save">  
    </form>
```

ERGEBNIS



1. Die Variablen \$stars und \$message werden initialisiert.
2. Die Variable \$star_ratings enthält ein indiziertes Array von Werten, die verwendet werden, um eine Gruppe von Optionsfeldern anzulegen.
3. Eine if-Anweisung überprüft, ob das Formular abgeschickt wurde.
4. Wenn ja, wird die ausgewählte Option aus dem superglobalen Array \$_POST ausgelesen.
5. Die PHP-Funktion in_array() prüft, ob der ausgewählte Wert zu den zulässigen Optionen zählt.
6. Mithilfe eines ternären Operators wird eine Meldung erstellt, die angibt, ob die Daten gültig waren oder nicht.
7. Der Wert in \$message wird angezeigt.
8. Eine foreach-Schleife erzeugt die Auswahlmöglichkeiten im HTML-Formular. Sie durchläuft die Werte im Array \$star_ratings. Für jeden dieser Werte geschieht Folgendes:
9. Die Auswahlmöglichkeit wird angezeigt, gefolgt von einem Optionsfeld, dessen value-Attribut die Option enthält.
10. Ein ternärer Operator überprüft, ob eine Option ausgewählt wurde. Trifft dies zu, wird das Attribut checked hinzugefügt.

SO ERKENNEN SIE, OB EIN KONTROLLFELD ANGEKREUZT WURDE

Ein Kontrollfeld kann entweder aktiviert oder deaktiviert sein, aber sein Name und Wert werden nur an den Server gesendet, wenn es aktiviert ist.

Um festzustellen, ob ein Kontrollfeld ausgewählt wurde oder nicht, sind zwei Schritte erforderlich:

- Zunächst überprüfen Sie mit der PHP-Funktion `isset()`, ob im superglobalen Array ein Wert für das Kontrollfeld vorhanden ist.
- Trifft dies zu, überprüfen Sie, ob der übermittelte Wert dem erwarteten Wert entspricht.

Beide dieser Überprüfungen lassen sich mit der Bedingung eines ternären Operators durchführen. Ergeben beide den Wert `true`, wissen Sie, dass der Benutzer das Kästchen angekreuzt hat, und Sie können den booleschen Wert `true` zuweisen.

Wenn beide Prüfungen den Wert `true` ergeben, wird in `$terms` der Wert `true` gespeichert, ansonsten `false`.

```
$terms = (isset($_POST['terms']) and $_POST['terms'] == true) ? true : false;
```

WENN DER WERT IN SUPERGLOBALES ARRAY ÜBERNOMMEN WURDE

UND DER BEREITGESTELLTE WERT GÜLTIG IST

Wenn dem Benutzer das Formular mit seiner markierten Auswahl erneut angezeigt wird, brauchen Sie nur zu überprüfen, ob der Wert für dieses Kontrollfeld wahr ist.

Trifft dies zu, wird dem Formularelement das Attribut `checked` hinzugefügt. Andernfalls wird stattdessen eine leere Zeichenkette hinzugefügt.

```
<input type="checkbox" name="terms" value="true"  
<?= $terms ? 'checked' : '' ?>
```

WENN DAS KONTROLLFELD AUSGEWÄHLT WURDE

HÄNGE ATTRIBUT Checked AN

ANSONSTEN HÄNGE LEEREN STRING AN

KONTROLLFELDER VALIDIEREN

PHP

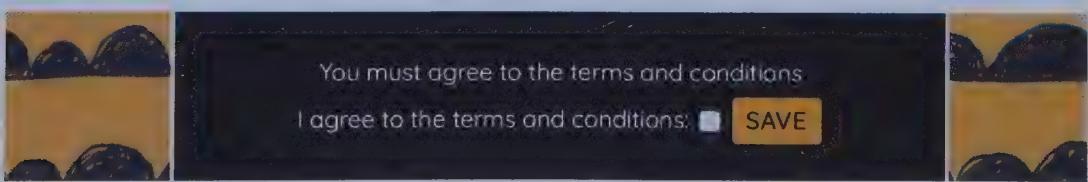
section_b/c06/validate-checkbox.php

```
<?php
① $terms  = '';
$messag = '';

② if ($_SERVER['REQUEST_METHOD'] == 'POST') {
③     $terms  = (isset($_POST['terms']) and $_POST['terms'] == true) ? true : false;
④     $message = $terms ? 'Thank you' : 'You must agree to the terms and conditions';
}
?> ...

⑤ <?= $message ?>
<form action="validate-checkbox.php" method="POST">
    I agree to the terms and conditions: <input type="checkbox" name="terms" value="true"
⑥     <?= $terms ? 'checked' : '' ?>>
    <input type="submit" value="Save">
</form>
```

ERGEBNIS



1. Die Variablen \$terms und \$message werden initialisiert.
2. Wenn das Formular abgeschickt wurde ...
3. Zwei Ausdrücke in der Bedingung eines ternären Operators dienen der Überprüfung, ob das Kontrollfeld angekreuzt wurde. Zunächst prüft die PHP-Funktion `isset()`, ob das Kontrollfeld gesendet wurde. Wenn dem so ist, überprüft der zweite Ausdruck, ob sein Wert `true` ist. Ergeben beide Ausdrücke den Wert `true`, bekommt die Variable \$terms ebenfalls den Wert `true` zugewiesen, andernsonst `false`.

4. Wenn \$terms den Wert `true` aufweist, werden in \$message die Worte `Thank you` gespeichert, andernfalls eine Nachricht, die den Benutzer auffordert, den Nutzungsbedingungen zuzustimmen.
5. Die Nachricht wird auf der Seite angezeigt.
6. Ein ternärer Operator überprüft, ob die Variable \$terms den Wert `true` enthält (was bedeutet, dass sie bereits validiert wurde). Ist dies der Fall, wird dem Kontrollfeld das Attribut `checked` hinzugefügt. Wenn nicht, wird stattdessen eine leere Zeichenkette hinzugefügt.

PRÜFEN, OB MEHRERE WERTE GÜLTIG SIND

Häufig müssen Seiten vor der Verarbeitung prüfen, ob eine Reihe von Daten gültig sind. Die meisten Formulare fordern beispielsweise zur Eingabe mehrerer Informationen auf.

Werfen Sie einen Blick auf das unten stehende Formular. Darin sollen die folgenden Daten eingegeben werden (unter Verwendung von drei Datentypen):

- Name (eine Zeichenkette zwischen 2 und 10 Zeichen lang)
- Alter (eine ganze Zahl zwischen 16 und 65)
- Zustimmung zu den Nutzungsbedingungen (ein boolescher Wert, der entweder true oder false lautet)

Wenn das Formular mit ungültigen Daten abgeschickt wird, sollte die Seite:

- die Daten nicht verarbeiten
- Fehlermeldungen erstellen, die dem Nutzer mitteilen, wie die einzelnen Probleme zu beheben sind
- alle vom Benutzer eingegebenen Werte anzeigen

The screenshot shows a dark-themed web page with a form for user input. The form fields are as follows:

- Name: Ivy (highlighted in red)
- Age: 15 (highlighted in red)
- I agree to the terms and conditions (checkbox)

Below the form, an error message is displayed: "Please correct the following errors:" followed by "Name: Ivy" and "Age: 15 You must be 16-65". At the bottom right of the form is a yellow "SAVE" button.

Zusätzlich zum Formular kann diese Seite auch Fehlermeldungen und alle bereits eingegebenen Werte anzeigen. Hierzu werden zunächst zwei Arrays deklariert:

- Eines mit einem Element für jeden der vom Anwender eingegebenen Werte.
- Eines mit einem Element, das die jeweiligen Fehlermeldungen enthält, die auf der Seite angezeigt werden können.

Beide Arrays müssen mit einem Namen für jedes Element und einem Wert initialisiert werden, der beim ersten Laden der Seite (noch vor dem Absenden des Formulars) angezeigt werden kann. Wird dies nicht beachtet und der PHP-Interpreter versucht, auf ein Element eines Arrays zuzugreifen, dem kein Wert zugeordnet ist, tritt ein Fehler auf.

Dem Array mit den Fehlermeldungen wird für jeden möglichen Formularfehler eine leere Zeichenkette zugewiesen, da beim ersten Laden der Seite noch keine Fehler auftreten.

- 1.** Nach dem Absenden des Formulars werden die eingegebenen Daten übernommen.

Diese Werte überschreiben die im Array für die Benutzerdaten gespeicherten Anfangswerte.

ARRAY FÜR BENUTZERDATEN

```
$user['name'] = $_POST['name'];
$user['age'] = $_POST['age'];
$user['terms'] = (isset($_POST['terms']) and $_POST['terms'] == true) ? true : false;
```

- 2.** Als Nächstes werden die einzelnen Daten validiert. Sind sie nicht gültig, wird eine Fehlermeldung im entsprechenden Element des Arrays \$errors gespeichert.

Die Daten werden mithilfe der Validierungsfunktionen überprüft, die Sie in diesem Kapitel kennengelernt haben und die true zurückgeben, wenn der vom Benutzer eingegebene Wert gültig ist, und false, wenn er ungültig ist.

Das heißt, dass die Validierungsfunktionen in der Bedingung eines ternären Operators aufgerufen werden können. Sind die Daten:

- Gültig: Dem Element wird eine leere Zeichenkette zugewiesen.
- Ungültig: Es wird eine Fehlermeldung gespeichert, die dem Benutzer mitteilt, warum die Daten ungültig sind.

ARRAY FÜR FEHLERQUELLEN

```
$errors['name'] = is_text($user['name'], 2, 20)
$errors['age'] = is_number($user['age'], 16, 65)
$errors['terms'] = $user['terms']
```

FORMULARWERTE VALIDIEREN

LEERE ZEICHENKETTE

FEHLERMELDUNG

```
? '' : 'Name must be 2-20 characters';
? '' : 'You must be 16-65';
? '' : 'You must agree to the terms';
```

- 3.** Um zu prüfen, ob Fehler vorliegen, werden alle Werte im Array \$errors mit der in PHP eingebauten Funktion implode() zu einem einzigen String zusammengefasst.

Das Ergebnis wird in der Variablen \$invalid gespeichert. Wenn \$invalid eine leere Zeichenkette enthält, waren die Daten gültig. Wenn nicht, gab es mindestens einen Fehler.

```
$invalid = implode($errors);
```

VARIABLE FÜR SÄMTLICHE FEHLERMELDUNGEN

ARRAY FÜR FEHLERQUELLEN

- 4.** Eine if-Anweisung überprüft, ob \$invalid einen Text enthält. Ist dies der Fall, wird dies als true gewertet, und die Fehlermeldungen werden zusammen mit dem Formular angezeigt.

Sind keine Fehler aufgetreten, enthält \$invalid einen leeren String (der als false behandelt wird), und die Seite kann die empfangenen Daten verarbeiten.

```
if ($invalid) {
    // Fehlermeldungen anzeigen und Daten nicht verarbeiten
} else {
    // Daten sind gültig, Seite kann Daten verarbeiten
}
```

FORMULARE VALIDIEREN

Dieses Beispiel zeigt, wie Sie mehrere Formularelemente validieren können. Das Ergebnis wurde bereits auf der vorherigen Seite gezeigt.

1. `validate.php` wird in die Seite eingebunden. Darin befinden sich die Definitionen für drei der in diesem Kapitel beschriebenen Validierungsfunktionen. Durch die Auslagerung dieser Funktionen in eine Include-Datei kann diese von jeder Seite aus eingebunden und die enthaltenen Funktionen genutzt werden.

2. Die Variable `$user` enthält ein Array mit einem Element für jedes Formularelement; diesen wird ein Anfangswert zugewiesen, der beim erstmaligen Laden der Seite im Formular verwendet wird.

3. Die Variable `$errors` enthält ein Array mit einem Element für jedes zu prüfende Datenelement.

4. `$message` wird eine leere Zeichenkette zugewiesen. Sobald die Daten validiert sind, steht darin eine Erfolgs- oder Fehlermeldung.

5. Eine `if`-Anweisung kontrolliert, ob das Formular abgeschickt wurde.

6. Wenn ja, werden die drei Angaben aus dem Formular erfasst, und diese Benutzereingaben überschreiben die im Array `$user` gespeicherten Anfangswerte.

7. Der eingegebene Name wird mit der Funktion `is_text()` validiert. Sie liefert `true` zurück, wenn die Daten gültig sind, und `false`, wenn nicht. Ist der Name gültig, wird im entsprechenden Element des Arrays `$errors` (siehe Schritt 3) ein leerer String abgelegt.

Wenn nicht, wird eine Meldung darin gespeichert, die dem Benutzer mitteilt, wie er das Problem beheben kann.

8. Das Alter des Benutzers wird mit `is_number()` überprüft.

Die Funktion liefert `true` zurück, wenn die Daten gültig sind, und `false`, wenn nicht. Ist das Alter gültig, wird im entsprechenden Element des Arrays `$errors` eine leere Zeichenkette abgelegt. Wenn nicht, wird darin eine Fehlermeldung gespeichert.

9. Wenn der Benutzer das Kontrollfeld für die Nutzungsbedingungen markiert hat, wird im entsprechenden Element des Arrays `$errors` ein leerer String abgelegt. Wenn nicht, wird darin eine Fehlermeldung gespeichert, die besagt, dass den Bedingungen zugesimmt werden muss.

10. Alle Werte im Array `$errors` werden mit der PHP-Funktion `implode()` zu einer einzigen Zeichenkette zusammengefügt. Das Ergebnis wird in der Variablen `$invalid` gespeichert.

11. Die Bedingung einer `if`-Anweisung überprüft, ob der Wert von `$invalid` `true` entspricht. Enthält die Variable einen beliebigen Text, so wird dieser als `true` gewertet. Eine leere Zeichenkette wird als `false` gewertet.

12. Wenn die Daten ungültig sind, wird in der Variablen `$message` eine Meldung gespeichert, die den Benutzer auffordert, die Formularfehler zu korrigieren.

13. Andernfalls enthält `$message` die Meldung, dass die Daten gültig waren. Wenn die Daten gültig sind, kann die Seite sie nachfolgend verarbeiten.

(Hat eine Seite gültige Daten erhalten, muss das Formular oftmals nicht mehr erneut angezeigt werden.)

14. Der in `$message` gespeicherte Wert wird angezeigt.

15. Wenn der Benutzer das Formular abgeschickt hat, wird der als Name eingegebene Wert in das `value`-Attribut des Formularelements geschrieben. (Dieser Text wird mit der PHP-Funktion `htmlspecialchars()` maskiert.)

Wurde das Formular nicht abgeschickt, wird die leere Zeichenkette angezeigt, die im entsprechenden Schlüssel des Arrays `$user` gespeichert wurde, als es in Schritt 2 initialisiert wurde.

16. Der Wert des diesem Formularelement entsprechenden Elements im Array `$errors` wird angezeigt.

17. Hat der Benutzer sein Alter angegeben, wird dieses im `value`-Attribut des Formularelements angezeigt. Danach folgt der entsprechende Wert aus dem Array `$errors`.

18. Hat der Besucher das Kontrollfeld für die Nutzungsbedingungen angekreuzt, wird diesem das Attribut `checked` hinzugefügt. Danach folgt der entsprechende Wert aus dem Array `$errors`.

```

<?php
declare(strict_types = 1); // Strikte Datentypen verwenden
① require 'includes/validate.php'; // Validierungsfunktionen

② $user = [
    'name' => '',
    'age'  => '',
    'terms' => ''
];
// Array $user initialisieren

③ $errors = [
    'name' => '',
    'age'  => '',
    'terms' => ''
];
// Array $errors initialisieren

④ $message = '';
// Nachricht initialisieren

⑤ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular gesendet wurde
    $user['name'] = $_POST['name']; // Namen abrufen
    $user['age']  = $_POST['age']; // Alter abrufen, Nutzungsbedingungen prüfen
    $user['terms'] = (isset($_POST['terms']) and $_POST['terms'] == true) ? true : false;
}

⑥ $errors['name'] = is_text($user['name'], 2, 20) ? '' : 'Must be 2-20 characters';
⑦ $errors['age']  = is_number($user['age'], 16, 65) ? '' : 'You must be 16-65';
⑧ $errors['terms'] = $user['terms'] ? '' : 'You must agree to the
    terms and conditions'; // Daten validieren

⑨
⑩ $invalid = implode($errors); // Fehlermeldungen zusammenführen
⑪ if ($invalid) { // Wenn es Fehler gab
    $message = 'Please correct the following errors:'; // nicht verarbeiten
    // ansonsten
}
⑫ } else { // Daten können verarbeitet werden
    $message = 'Your data was valid';
}
⑬
⑭ ?> ...
⑮ <?= $message ?>
<form action="validate-form.php" method="POST">
⑯ Name: <input type="text" name="name" value="<?= htmlspecialchars($user['name']) ?>">
⑰ <span class="error"><?= $errors['name'] ?></span><br>
⑱ Age: <input type="text" name="age" value="<?= htmlspecialchars($user['age']) ?>">
⑲ <span class="error"><?= $errors['age'] ?></span><br>
⑳ <input type="checkbox" name="terms" value="true" <?= $user['terms'] ? 'checked' : '' ?>>
    I agree to the terms and conditions
    <span class="error"><?= $errors['terms'] ?></span><br>
    <input type="submit" value="Save">
</form>

```

FILTERFUNKTIONEN ZUR DATENERFASSUNG

PHP verfügt auch über zwei eingebaute Funktionen, die die vom Browser gesendeten Daten erfassen und sie in Variablen speichern. Sie werden Filterfunktionen genannt, weil sie einen Filter auf die vom Browser gesendeten Daten anwenden können.

`filter_input()` ruft einen einzelnen an den Server übermittelten Wert ab. Die Funktion erfordert zwei Argumente. Das erste Argument ist die Eingabequelle (die nicht in Anführungszeichen geschrieben wird).

Nutzen Sie:

- `INPUT_GET`, um über HTTP GET gesendete Daten zu erhalten
- `INPUT_POST`, um über HTTP POST gesendete Daten zu erhalten
- `INPUT_SERVER`, um dieselben Daten zu erhalten, die im superglobalen Array `$_SERVER` verfügbar gemacht wurden

```
$data = filter_input(INPUT_SOURCE, 'name');
```

[] []
EINGABEQUELLE NAME

`filter_input_array()` sammelt alle Werte, die per HTTP GET oder POST an den Server gesendet wurden, und speichert jeden als einzelnes Element eines Arrays.

```
$data = filter_input_array(INPUT_SOURCE);
```

[]
EINGABEQUELLE

Die empfangenen Daten werden als String-Datentyp gespeichert. Einige der von diesen Filterfunktionen bereitgestellten Filter konvertieren den Datentyp.

Das zweite Argument ist der Name eines an den Server übermittelten Name/Wert-Paars und sollte in Anführungszeichen stehen. So verwendet, gibt `filter_input()` Folgendes zurück:

- den Wert, falls er an den Server gesendet wurde
- `null`, wenn die Daten nicht an den Server übermittelt wurden

Nachdem Sie gelernt haben, diese Funktion anzuwenden, erfahren Sie außerdem, wie Sie als dritten Parameter einen Filter einsetzen können.

Da sie alle Werte abruft, benötigt sie nur ein Argument: die Eingabequelle. Die Werte für die Eingabequelle sind dieselben wie bei `filter_input()`.

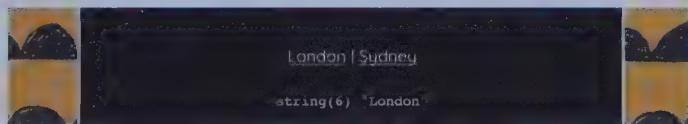
Auf den nächsten Seiten nutzen wir die PHP-Funktion `var_dump()`, um die mit diesen Funktionen erfassten Werte anzuzeigen, da es wichtig ist, die Datentypen der einzelnen Werte zu betrachten.

DATEN ABFRAGEN MIT FILTERFUNKTIONEN

PHP

```
① <?php $location = filter_input(INPUT_GET, 'city'); ?> ...
② [a href="filter_input.php?city=London">London</a> | 
  a href="filter_input.php?city=Sydney">Sydney</a>
③ <pre><?php var_dump($location); ?></pre>
```

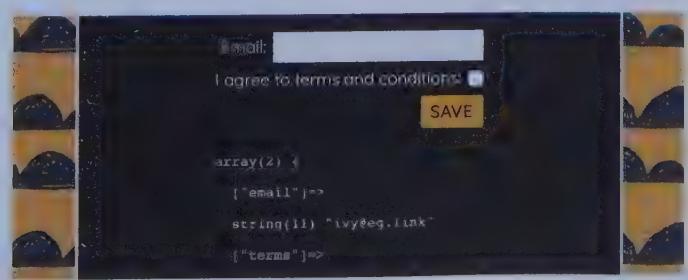
ERGEBNIS



PHP

```
④ <?php $form = filter_input_array(INPUT_POST); ?> ...
<form action="filter_input_array.php" method="POST">
  Email: <input type="text" name="email" value=""><br>
  I agree to terms and conditions:
  <input type="checkbox" name="terms" value="true"><br>
  <input type="submit" value="Save">
</form>
⑥ <pre><?php var_dump($form); ?></pre>
```

ERGEBNIS



Wenn diese beiden Beispiele zum ersten Mal geladen werden, ist der angezeigte Wert NULL, da der Query-String leer ist.

1. Die Funktion `filter_input()` erfasst einen im Query-String per HTTP GET gesendeten Einzelwert. Der Name des Name/Wert-Paars ist `city`. Der erfasste Wert wird in der Variablen `$location` gespeichert.

2. Zwei Links übertragen mit Query-Strings jeweils einen Namen mit der Bezeichnung `city`; die Werte sind unterschiedliche Städtenamen.

3. Mittels `var_dump()` wird der in `$location` gespeicherte Wert und sein Datentyp (String) angezeigt.

4. Mithilfe der Funktion `filter_input_array()` werden alle Werte aus dem Formular abgerufen, wenn es mittels HTTP POST übertragen wird. Das von dieser Funktion erstellte Array wird in der Variablen `$form` gespeichert.

5. Das Formular versendet ein Texteingabefeld und ein Kontrollfeld über HTTP POST.

6. `var_dump()` zeigt die in `$form` gespeicherten Namen und Werte zusammen mit dem jeweiligen Datentyp an.

Probieren Sie es: Senden Sie das Formular ab, ohne es auszufüllen. Das Array enthält für das Texteingabefeld eine leere Zeichenkette und für das Kontrollfeld gar nichts.

FILTER ZUR DATENVALIDIERUNG

Wenn Filterfunktionen vom Browser gesendete Daten abfragen, speichern sie diese als Zeichenkette. Unten sehen Sie drei Validierungsfilter, die überprüfen, ob es sich bei einem Wert um einen Boolean, eine Ganzzahl oder eine Fließkommazahl handelt. Jeder Filter ist durch eine eigene Filter-ID gekennzeichnet.

Wenn eine Seite einen booleschen, ganzzahligen oder fließenden Wert erwartet, können die Filterfunktionen anhand der drei nachstehenden Filter prüfen, ob der bereitgestellte Wert dem richtigen Datentyp entspricht.

Bei der Überprüfung, ob es sich bei einem Wert um einen booleschen Wert, eine Ganzzahl oder eine Fließkommazahl handelt, konvertieren die Filterfunktionen den Wert vom Datentyp String in den im Filter angegebenen Datentyp. Wie Sie die Filter einsetzen, erfahren Sie auf Seite 273.

FILTER-ID	BESCHREIBUNG
FILTER_VALIDATE_BOOLEAN	Überprüft, ob ein Wert true ist. 1, on und yes zählen alle als true. Groß- und Kleinschreibung werden nicht unterschieden. Entspricht der Wert true, gibt die Funktion den booleschen Wert true zurück. Wenn nicht, gibt sie false zurück.
FILTER_VALIDATE_INT	Prüft, ob eine Zahl eine Ganzzahl ist (0 wird nicht als gültige Ganzzahl betrachtet). Falls dies zutrifft, wird die Zahl als int-Datentyp zurückgegeben. Wenn nicht, gibt die Funktion false zurück.
FILTER_VALIDATE_FLOAT	Prüft, ob eine Zahl eine Fließkommazahl (Dezimalbruch) ist. Ganzahlen bestehen die Prüfung (0 nicht, da sie nicht als gültige Ganzzahl gewertet wird). Trifft dies zu, wird der Zahlenwert als float-Datentyp zurückgegeben. Wenn nicht, gibt die Funktion false zurück.

Jeder Filter verfügt außerdem über zwei Arten von Einstellungen, mit denen Sie sein Verhalten steuern können:

- **Flags sind Einstellungen, die Sie ein- oder ausschalten können.**
- **Optionen sind Einstellungen, für die Sie einen Wert festlegen müssen.**

Die Integer- und Float-Filter bieten beispielsweise Optionen zur Eingabe eines Mindest- und Höchstwerts, den der Benutzer angeben darf. Wenn also ein Besucher nach seinem Alter gefragt wird und zwischen 16 und 65 Jahre alt sein muss, könnte der Filter prüfen, ob die angegebene Zahl in diesem Bereich liegt.

Wurde keine Zahl eingegeben oder war die Zahl zu niedrig oder zu hoch, wäre sie ungültig.

Alle Validierungsfilter bieten auch die Möglichkeit, einen Standardwert anzugeben. Dieser wird herangezogen, wenn die abgerufenen Daten ungültig sind.

Flags sind Optionen, die lediglich aktiviert werden können.

Der Integer-Filter hat beispielsweise ein Flag, das Sie aktivieren können, um Besuchern die Möglichkeit zu geben, neben den Standardziffern 0–9 auch Hexadezimalzahlen anzugeben. (Die Hexadezimalschreibweise stellt mit den Ziffern 0–9 und den Buchstaben A–F die Zahlen 10–15 dar; Sie kennen sie vielleicht von den Farbangaben in HTML und CSS.)

Eine umfassende Liste der Validierungsfilter mit ihren Flags und Optionen finden Sie auf den Seiten 212–215.

Unten sehen Sie die Validierungsfilter zum Erfassen von Text. Mit regulären Ausdrücken sind auch benutzerdefinierte Filter möglich.

Daten unterliegen oft bestimmten Regeln:

- zulässige Zeichenanzahl
- welche Zeichen zulässig sind
- Reihenfolge, in der diese Zeichen erscheinen müssen

So gibt es beispielsweise Regeln, zur Verwendung von Zeichen in E-Mail-Adressen, URLs, Domain-Namen und IP-Adressen. Die vier nachstehenden Filter überprüfen, ob ein Wert diesen Regeln entspricht.

FILTER-ID	BESCHREIBUNG
FILTER_VALIDATE_EMAIL	Prüft, ob die Zeichenkettenstruktur einer E-Mail-Adresse entspricht.
FILTER_VALIDATE_URL	Prüft, ob die Zeichenkettenstruktur einer URL entspricht.
FILTER_VALIDATE_DOMAIN	Prüft, ob die Zeichenkettenstruktur einem gültigen Domain-Namen entspricht.
FILTER_VALIDATE_IP	Prüft, ob die Zeichenkettenstruktur einer gültigen IP-Adresse entspricht.

Mit regulären Ausdrücken lassen sich andere Filter schreiben, die prüfen, ob ein Wert ein bestimmtes Zeichenmuster enthält.

Der reguläre Ausdruck wird als Option des Filters FILTER_VALIDATE_REGEXP angegeben.

FILTER-ID	BESCHREIBUNG
FILTER_VALIDATE_REGEXP	Prüft, ob eine Zeichenkette ein Zeichenmuster enthält, das durch einen regulären Ausdruck beschrieben wird (siehe Seiten 282–283).

EINZELWERTE MIT FILTERN VALIDIEREN

Wenn Daten mithilfe der Filterfunktionen geprüft werden, müssen die ID des zu verwendenden Filters und alle Flags oder Optionen, nach denen sich der Filter richten soll, übergeben werden.

Wenn `filter_input()` verwendet wird, um ein einzelnes Datenelement zu erfassen, ist der dritte Parameter die ID des zu verwendenden Filters, und der vierte (optionale) Parameter enthält die Einstellungen, die der Filter verwenden kann.

Die Funktion liefert:

- den Wert, den sie bekommen hat, falls der Filter bestanden wurde
 - `false`, wenn die Daten den Filter nicht bestehen
 - `null`, wenn der Name nicht an den Server gesendet wurde

```
$data = filter_input(INPUT_SOURCE, 'name', FILTER_ID[, $settings]);
```

EINGABE-QUELLE	NAME	ID	FLAGS/OPTIONEN
----------------	------	----	----------------

Wenn ein Filter Flags und Optionen verwendet, werden diese in einem assoziativen Array mit zwei Schlüsseln gespeichert:

- flags enthält Einstellungen, die eingeschaltet werden können
 - options enthält Einstellungen, die einen Wert erfordern

Nachfolgend wird ein Array aus Flags und Optionen in der Variablen \$settings gespeichert.

Der Wert für den Schlüssel `f1ags` ist der Name des zu aktivierenden Flags (dieser wird nicht in Anführungszeichen geschrieben). Um mehrere Flags zu verwenden, trennen Sie deren Namen jeweils mit dem Pipe-Symbol `|`.

Der Wert für den Schlüssel `options` ist ein weiteres assoziatives Array; der Schlüssel jedes Elements ist der Name der zu setzenden Option, und der Wert ist der zu verwendende Wert.

```
$settings['flags'] = FLAG_NAME1 | FLAG_NAME2;  
$settings['options']['option1'] = value1;  
$settings['options']['option2'] = value2;
```

OPTION WERT

Wenn ein Element eines Arrays ein anderes Array speichert, ist die obige Syntax möglicherweise einfacher lesbar. Dieser Ansatz wurde auf Seite 42 als Methode zur Aktualisierung von Arrays vorgestellt.

Hinweis: Wenn `filter_input()` ungültige Daten entgegennimmt, wird `false` zurückgegeben. Das bedeutet, dass der vom Benutzer eingegebene Wert nicht in einem Formular angezeigt werden kann.

WERTE MIT FILTERN ERFASSEN

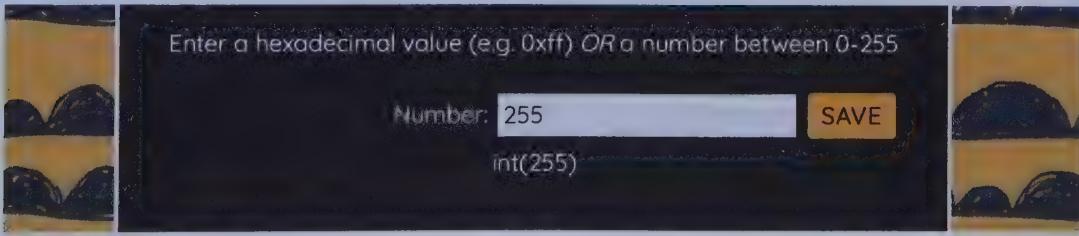
PHP

section_b/c06/validate-input.php

```
<?php
$settings['flags']           = FILTER_FLAG_ALLOW_HEX; // Hex-Zahlen erlauben (Flag)
① $settings['options']['min_range'] = 0;                  // Mindestwert (Option)
$settings['options']['max_range'] = 255;                 // Maximalwert (Option)

② $number = filter_input(INPUT_POST, 'number', FILTER_VALIDATE_INT, $settings);
?> ...
<form action="validate-input.php" method="POST">
③ Number: <input type="text" name="number" value=<?= htmlspecialchars($number) ?>>
<input type="submit" value="Save">
</form>
④ <?php var_dump($number); ?>
```

ERGEBNIS



1. Die Variable `$settings` enthält ein Array aus Flags und Optionen, die zur Validierung einer Zahl verwendet werden. Das Flag gestattet die Angabe einer Zahl in Hexadezimalschreibweise. Die Optionen geben an, dass die Zahl zwischen 0 und 255 liegen muss.

2. Die Funktion `filter_input()` ruft den per HTTP POST gesendeten Wert eines Formularelements ab, dessen Name Zahl ist. Der dritte Parameter enthält die Filter-ID. Der vierte Parameter entspricht dem Namen der Variablen, die das Array von Optionen und Flags enthält, die für den Filter verwendet werden sollen.

3. Der in `$number` gespeicherte Wert wird im Formularelement angezeigt. Wenn der Wert in `$number` null ist (weil das Formular nicht abgeschickt wurde) oder false (weil die Daten ungültig waren), wird er im Formularelement nicht angezeigt. Das liegt daran, dass PHP für die Werte false und null nichts anzeigt.

4. Mittels `var_dump()` wird der in `$number` gespeicherte Wert angezeigt (da false oder null im Browser nicht angezeigt werden). Außerdem wird der Datentyp angezeigt, da alle gültigen Zahlen von Zeichenketten in Ganzzahlen umgewandelt werden.

FILTER ZUR VALIDIERUNG MEHRERER EINGABEWERTE

Um eine Reihe von Werten gleichzeitig zu erfassen und zu validieren, können Sie `filter_input_array()` verwenden und für jede erfasste Dateneinheit einen eigenen Filter angeben.

Wenn eine Seite mehrere Werte erwartet, erstellen Sie ein assoziatives Array mit einem Element für jeden Wert, dessen Empfang die Seite erwartet. Die Schlüssel für die einzelnen Array-Elemente sind die Namen der Formularelemente oder die Namen der Query-String.

Der Wert der einzelnen Elemente ist entweder:

- der Name des Filters, der beim Erfassen der Daten verwendet werden soll (wenn er keine Flags oder Optionen hat), oder
- ein Array, das den Namen des Filters und alle zu verwendenden Flags oder Optionen enthält.

```
$filters['name1'] = FILTER_ID;  
$filters['name2']['filter'] = FILTER_ID;  
$filters['name2']['options']['option1'] = value1;  
$filters['name2']['options']['option2'] = value2;
```

Die Funktion `filter_input_array()` wird dann mit zwei Parametern aufgerufen:

- Eingabekette (`INPUT_GET` oder `INPUT_POST`)
- Array aus Filtern, die für die Werte verwendet werden sollen, deren Erhalt die Seite für die jeweiligen Eingaben erwartet

Sie liefert ein neues assoziatives Array zurück. Der Schlüssel jedes Elements ist der Name der Eingabe; der Wert ist:

- der übermittelte Wert, falls er gültig ist
- `false`, wenn der Wert zwar übermittelt wurde, aber ungültig ist
- `null`, wenn der Name nicht übermittelt wurde

```
$data = filter_input_array(INPUT_SOURCE, $filters);
```

EINGABE-QUELLE FILTER-ARRAY

Wenn die Seite zusätzliche Daten erhält, die nicht im Filter-Array angegeben wurden, werden diese Daten nicht zum Array hinzugefügt, das `filter_input_array()` zurückgibt.

Wenn ein Teil der Daten fehlt, wird diesem der Wert `null` zugewiesen. Um zu verhindern, dass dem Array fehlende Werte hinzugefügt werden, geben Sie als drittes Argument `false` an.

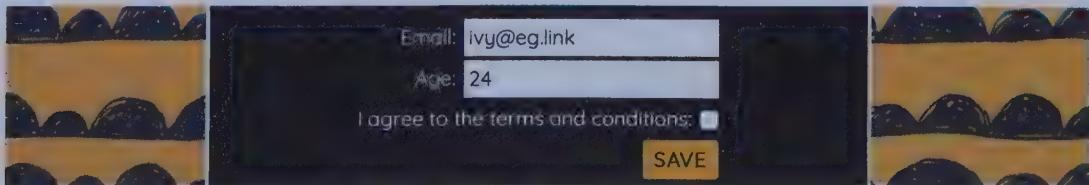
MEHRERE EINGABEN MIT FILTERN VALIDIEREN

PHP

section_b/c06/validate-multiple-inputs.php

```
<?php  
① $form['email'] = '';  
    $form['age'] = '';  
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
        ② $filters['email'] = FILTER_VALIDATE_EMAIL; // E-Mail initialisieren  
        $filters['age']['filter'] = FILTER_VALIDATE_INT; // Alter initialisieren  
        $filters['age']['options']['min_range'] = 16; // Wenn übermittelt  
        ③ $form = filter_input_array(INPUT_POST, $filters); // Ganzzahl-Filter  
    } // Mindestalter 16  
    // Daten validieren  
?  
...  
<form action="validate-multiple-inputs.php" method="POST">  
    Email: <input type="text" name="email" value="<?= htmlspecialchars($form['email']) ?>">  
    Age: <input type="text" name="age" value="<?= htmlspecialchars($form['age']) ?>"><br>  
    I agree to the terms and conditions: <input type="checkbox" name="terms" value="1"><br>  
    <input type="submit" value="Save">  
</form>  
④ <pre><?php var_dump($form); ?></pre>
```

ERGEBNIS



1. Das Array `$form` wird mit Werten für die Eingabefelder `email` und `age` initialisiert.

2. Wenn das Formular abgeschickt wurde, enthält `$filters` ein Array. Der Schlüssel eines jeden Elements ist der Name eines Formularelements. Die Werte sind die zu verwendenden Filter/Optionen:
- `email` muss dem Format einer E-Mail Adresse entsprechen.
 - `age` muss eine ganze Zahl größer oder gleich 16 sein.

3. `filter_input_array()` erfasst und überprüft die Daten und überschreibt die in `$form` gespeicherten Werte.

4. Die Funktion `var_dump()` zeigt die Daten an.

Hinweis: Beim Absenden des Formulars wird das Kontrollfeld auch dann nicht zum Array `$form` hinzugefügt, wenn es aktiviert ist, da es im Array `$filters` nicht aufgeführt wurde. Außerdem werden ungültige Daten nicht im Formularelement angezeigt.

FILTERFUNKTIONEN ZUM UMGANG MIT VARIABLEN

PHP bietet zwei integrierte Filterfunktionen zur Filterung von in Variablen gespeicherten Werten. `filter_var()` wendet einen Filter auf einen in einer Variablen gespeicherten Einzelwert an. `filter_var_array()` wendet Filter auf eine Gruppe von Werten in einem Array an.

Die Funktion `filter_var()` benötigt:

- den Namen der Variablen, deren Wert geprüft werden soll
- eine Filter-ID

Die Werte für Optionen oder Flags werden genauso gesetzt wie bei `filter_input()`. Die Rückgabewerte sind ebenfalls identisch: Ist der Wert gültig, wird er zurückgeliefert; ist er ungültig, wird `false` zurückgeliefert; fehlt er, wird `null` zurückgeliefert.

`filter_var($variable, FILTER_ID[, $settings]);`

VARIABLE MIT DATEN

FILTER

FLAGS/OPTIONEN

Die Funktion `filter_var_array()` weist ebenfalls zwei Parameter auf:

- Name der Variablen, in der sich das Array mit den zu prüfenden Daten befindet
- Array mit Filtern und deren Optionen/Flags

Die Werte für die Optionen oder Flags werden auf die gleiche Weise gesetzt wie bei `filter_input_array()`, und auch die Rückgabewerte sind gleich.

Wird nur ein Filter angegeben, wird dieser auf alle Werte im Array angewendet.

`filter_var_array($array, $filters);`

VARIABLE MIT ARRAY

EINZUSETZENDE FILTER

Wenn die Funktionen `filter_input()` oder `filter_input_array()` zur Datenüberprüfung eingesetzt werden, geben sie für ungültige Daten immer `false` zurück (und ersetzen damit den vom Benutzer eingegebenen Wert).

Enthält das Formular also ungültige Daten, kann der Benutzer seine ungültigen Eingaben im Formular nicht sehen.

Um falsche Eingaben anzuzeigen, rufen Sie die Daten ab und speichern sie in einer Variablen oder einem Array. Wenn sie dann mit `filter_var()` oder `filter_var_array()` überprüft werden, lässt sich das Ergebnis in einer neuen Variablen speichern. Wenn die Daten:

- gültig sind: Die Seite verwendet die Daten aus der neuen Variablen
- ungültig sind: Das Formular zeigt die Daten so an, wie sie ursprünglich vor der Validierung erfasst wurden

DATEN IN VARIABLEN VALIDIEREN

PHP

section_b/c06/validate-variables.php

```
<?php
    $form['email'] = '';
    $form['age']   = '';
    $form['terms'] = 0;
    $data          = [];

    ① if ($_SERVER['REQUEST_METHOD'] == 'POST') {           // Wenn übermittelt
        $filters['email']           = FILTER_VALIDATE_EMAIL; // E-Mail-Filter
        $filters['age']['filter']    = FILTER_VALIDATE_INT;   // Ganzzahl-Filter
        $filters['age']['options']['min_range'] = 16;         // Mindestalter
        $filters['terms']           = FILTER_VALIDATE_BOOLEAN; // Boolean-Filter
    ② $form = filter_input_array(INPUT_POST);               // Alle Werte abrufen
    ③ $data = filter_var_array($form, $filters);            // Filter anwenden
    }

    ?> ...

<form action="validate-variables.php" method="POST">
    ④ Email: <input type="text" name="email" value="<?= htmlspecialchars($form['email']) ?>">
    Age: <input type="text" name="age" value="<?= htmlspecialchars($form['age']) ?>"><br>
    I agree to the terms and conditions: <input type="checkbox" name="terms" value="1"><br>
    <input type="submit" value="Save">
</form>
⑤ <pre><?php var_dump($data); ?></pre>
```

Dieses Beispiel sieht genauso aus wie das vorherige.

1. Die Arrays `$form` und `$data` werden mit Werten initialisiert, um anzusehen, dass das Formular noch nicht abgeschickt wurde.
2. Wenn das Formular abgeschickt wurde, werden im Array `$filters` die Filter und Optionen zur Datenüberprüfung gespeichert.
3. `filter_input_array()` erfasst die Daten aus dem Formular und überschreibt die in Schritt 1 in `$form` gespeicherten Werte.
4. `filter_var_array()` überprüft die Formulardaten (unter Verwendung der im Array `$filters` angegebenen Filter). Die Funktion speichert das Ergebnis-Array dann in der Variablen `$data`.

5. Die Texteingabefelder zeigen die vom Benutzer eingegebenen Werte an (die vor der Datenvalidierung im Array `$form` gespeichert wurden).

6. Die Funktion `var_dump()` zeigt die Daten an, die validiert und im Array `$data` gespeichert wurden (oder `null`, wenn das Formular nicht abgeschickt wurde).

Probieren Sie es: Entfernen Sie das Eingabefeld für das Alter und senden Sie das Formular erneut ab. Die Funktion `var_dump()` zeigt in Schritt 6 immer noch einen Wert für das Steuerelement `age` an, weil es im Filter-Array zwar aufgeführt war, beim Aufruf von `filter_var_array()` jedoch nicht übergeben wurde.

VALIDIERUNGSFILTER, FLAGS UND OPTIONEN

Die Tabellen unten zeigen die Filter, Flags und Optionen zur Verarbeitung von booleschen Werten und Zahlen. Auf der rechten Seite finden Sie Filter, Flags und Optionen für den Umgang mit Zeichenketten.

Alle Validierungsfilter haben auch die Option `default`.

Mit dieser Option können Sie einen Standardwert angeben, der eingesetzt wird, falls die Daten ungültig sind.

FILTER_VALIDATE_BOOLEAN

Prüft, ob ein Wert `true` ist (`1`, `on` oder `yes` werden als `true` behandelt). Gibt in diesem Fall den booleschen Wert `true` zurück und ansonsten `false`. Ist der Name nicht vorhanden, wird `null` zurückgegeben. Groß- und Kleinschreibung werden nicht unterschieden.

FILTER_VALIDATE_INT

Prüft, ob eine Zahl eine Ganzzahl ist (`0` wird nicht als gültige Ganzzahl gezählt). Eine gültige Ganzzahl wird als Datentyp `int` zurückgegeben.

FILTER_VALIDATE_FLOAT

Prüft, ob eine Zahl eine Gleitkommazahl (Dezimalbruch) ist. Ganzzahlen sind ebenfalls gültig (`0` wird jedoch nicht als gültige Ganzzahl betrachtet). Ein gültiger Wert wird als Datentyp `float` zurückgegeben.

FLAG	BESCHREIBUNG
<code>FILTER_NULL_ON_FAILURE</code>	Gibt <code>null</code> zurück (nicht <code>false</code>), wenn ungültig

FLAG	BESCHREIBUNG
<code>FILTER_FLAG_ALLOW_HEX</code>	Hexadezimalzahlen erlauben
<code>FILTER_FLAG_ALLOW_OCTAL</code>	Oktalzahlen erlauben

OPTION	BESCHREIBUNG
<code>min_range</code>	zulässiger Mindestwert
<code>max_range</code>	zulässiger Höchstwert

FLAG	BESCHREIBUNG
<code>FILTER_FLAG_ALLOW_THOUSAND</code>	Erlaubt Fließkommazahlen mit Tausender-trennzeichen Gibt <code>null</code> zurück (nicht <code>false</code>), wenn ungültig

FILTER_VALIDATE_REGEXPC

Prüft, ob eine Zeichenkette ein Zeichenmuster enthält, das in einem regulären Ausdruck definiert ist (siehe Seiten 214 bis 217).

OPTION	BESCHREIBUNG
regexp	zu betrachtender regulärer Ausdruck

FILTER_VALIDATE_EMAILC

Prüft, ob die Struktur einer Zeichenkette einer E-Mail-Adresse entspricht.

FLAG	BESCHREIBUNG
FILTER_FLAG_EMAIL_UNICODE	Erlaubt Unicode-Zeichen im Namensteil der Adresse (vor dem @-Symbol)

FILTER_VALIDATE_URLC

Prüft, ob die Struktur einer Zeichenkette einer gültigen URL entspricht.

FLAG	BESCHREIBUNG
FILTER_FLAG_SCHEME_REQUIRED	Muss ein Adressschema enthalten z.B. <code>http://</code> oder <code>ftp://</code>
FILTER_FLAG_HOST_REQUIRED	Muss einen Hostnamen enthalten
FILTER_FLAG_PATH_REQUIRED	Muss einen Pfad oder ein Verzeichnis enthalten
FILTER_FLAG_QUERY_REQUIRED	Muss einen Query-String enthalten

FILTER_VALIDATE_DOMAINC

Prüft, ob die Struktur einer Zeichenkette einem Domain-Namen entspricht.

FLAG	BESCHREIBUNG
FILTER_FLAG_HOSTNAME	Validiert einen Hostnamen

FILTER_VALIDATE_IPC

Prüft, ob die Struktur einer Zeichenkette einer gültigen IP-Adresse entspricht.

FLAG	BESCHREIBUNG
FILTER_FLAG_IPV4	Prüft, ob es sich um eine gültige IPv4-IP-Adresse handelt
FILTER_FLAG_IPV6	Prüft, ob es sich um eine gültige IPv6-IP-Adresse handelt
FILTER_FLAG_NO_RES_RANGE	Lässt keine IPs aus reservierten Bereichen zu (Adressen, die in lokalen Netzwerken verwendet und nicht über das Internet gesendet werden)
FILTER_FLAG_NO_PRIV_RANGE	Erlaubt keine IPs aus dem privaten Bereich (einer Teilmenge der reservierten IP-Adressen)

FILTER ZUR DATENBEREINIGUNG

Bei der Datenbereinigung werden Zeichen entfernt (oder optional auch ersetzt), die nicht in einem Wert enthalten sein dürfen. Alle vier PHP-Filterfunktionen können auf eine Reihe integrierter Filter zurückgreifen, um Daten zu bereinigen.

Zusätzlich zu den Validierungsfiltern können die in PHP eingebauten Filterfunktionen auch verschiedene integrierte Bereinigungsfilter anwenden, die Zeichen entfernen (und manchmal auch ersetzen), die nicht in einem Wert vorkommen sollten.

Der erste Filter in der unten stehenden Tabelle erfüllt die gleiche Aufgabe wie die Funktion `htmlspecialchars()` (Seite 246); er ersetzt die fünf von HTML als Code behandelten reservierten Zeichen durch Entitäten.

Der zweite Filter kodiert eine URL, indem er die Zeichen, die nicht in einer URL erscheinen dürfen, durch kodierte Versionen dieser Zeichen ersetzt.

Die übrigen Filter entfernen Zeichen, die in Text, Zahlen, E-Mail-Adressen und URLs nicht vorkommen dürfen (sie ersetzen diese Zeichen jedoch nicht).

Sie sollten Daten maskieren oder bereinigen, wenn sie verwendet werden (und nicht, wenn sie erfasst werden), denn dadurch werden die Daten verändert. Stellen Sie sich zum Beispiel vor, ein Besucher gibt den Text `Fish & Chips` ein.

Um diesen Text auf einer Seite anzuzeigen, muss das kaufmännische Und maskiert werden: `Fish & Chips`. Wurde der Text jedoch bei der Erfassung maskiert, kann eine Suchfunktion den Text `Fish & Chips` möglicherweise nicht finden, da das kaufmännische Und bereits maskiert ist.

Außerdem werden Zeichen je nach Verwendung der Daten auf unterschiedliche Weise maskiert; dies wird als Nutzungskontext der Daten bezeichnet. Um denselben Text in einem Query-String anzuzeigen, wird das Leerzeichen durch `%20` und das kaufmännische Und durch `%26` ersetzt, sodass daraus Folgendes wird:

`http://eg.link/search.php?Fish%20%26%20Chips`

FILTER-ID	BESCHREIBUNG
<code>FILTER_SANITIZE_FULL_SPECIAL_CHARS</code>	Entspricht <code>htmlspecialchars()</code> mit aktiviertem ENT_QUOTES
<code>FILTER_SANITIZE_ENCODED</code>	Konvertiert URL in eine URL-kodierte Version derselben URL
<code>FILTER_SANITIZE_STRING</code>	Entfernt Tags aus einem String
<code>FILTER_SANITIZE_NUMBER_INT</code>	Entfernt alle Zeichen außer 0–9 und + oder -
<code>FILTER_SANITIZE_NUMBER_FLOAT</code>	Entfernt alle Zeichen außer 0–9 und + oder -. Besitzt Flags für Tausender- und Dezimaltrennzeichen und um e oder E für wissenschaftliche Notation zu erlauben.
<code>FILTER_SANITIZE_EMAIL</code>	Entfernt unzulässige Zeichen aus E-Mail-Adressen. Erlaubt sind A-z 0-9 !#\$%&'*+-=?^_`{ } ~@.[]
<code>FILTER_SANITIZE_URL</code>	Entfernt unzulässige Zeichen aus URLs. Erlaubt sind A-z 0-9 \$-_+.!*()'{} \ ^~[]`<>#%" ;/?:@&=

BEREINIGUNGSFILTER AUF VARIABLEN ANWENDEN

PHP

section_b/c06/sanitization-filters.php

```
<?php  
① $user['name'] = 'Ivy<script src="js/bad.js"></script>';           // Name,  
① $user['age']  = 23.75;                                              // Alter,  
① $user['email'] = 'ivy@eg.link/';                                     // E-Mail des Nutzers  
  
② $sanitize_user['name'] = FILTER_SANITIZE_FULL_SPECIAL_CHARS; // HTML-Escape-Filter  
② $sanitize_user['age']  = FILTER_SANITIZE_NUMBER_INT;      // Integer-Filter  
② $sanitize_user['email'] = FILTER_SANITIZE_EMAIL;          // E-Mail-Filter  
  
③ $user = filter_var_array($user, $sanitize_user);           // Ausgabe bereinigen  
    ?> ...  
④ <p>Name: <?= $user['name'] ?></p>  
④ <p>Age:   <?= $user['age'] ?></p>  
④ <p>Email: <?= $user['email'] ?></p>  
⑤ <pre><?php var_dump($user); ?></pre>
```

1. Die Variable `$user` enthält ein Array mit Benutzerdaten.

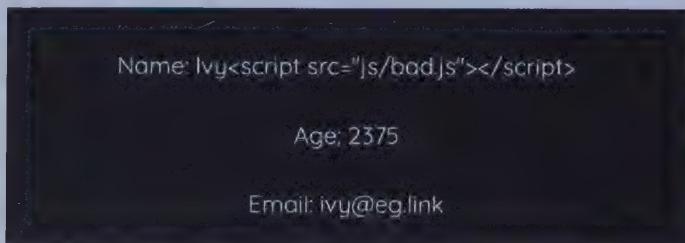
2. Die Variable `$sanitize_user` enthält ein Array mit drei Schlüsseln, deren Namen mit den Schlüsseln im Array `$user` übereinstimmen. Die Werte sind die Namen der Bereinigungsfilter, die für diesen Wert verwendet werden sollen.

3. Die Funktion `filter_var_array()` wird aufgerufen, um die Bereinigungsfilter auf die im Array `$user` gespeicherten Werte anzuwenden.

Der Name wird maskiert, und unerwünschte Zeichen werden aus der Altersangabe und der E-Mail-Adresse entfernt.

4. Die bereinigten Daten werden angezeigt.

ERGEBNIS



5. Die PHP-Funktion `var_dump()` zeigt das bereinigte Array `$user` an (im Ergebnis oben nicht zu sehen).

Probieren Sie es: Entfernen Sie in Schritt 2 das Element `age` aus dem Array `$user`. Da es im Filter-Array benannt wurde, erhält es im Array `$user` den Wert `null`.

Hinweis: Das Dezimaltrennzeichen wurde aus der Altersangabe entfernt, sie lautet 2375. Achten Sie darauf, dass die Datenbereinigung Ihre Werte nicht verändert. Um Dezimal- oder Tausender-trennzeichen oder die wissenschaftliche Notation zuzulassen, verwenden Sie den NUMBER-Filter mit Flags: <http://notes.re/php/sanitize>.

FORMULARE MIT FILTERN VALIDIEREN

In diesem Beispiel werden Validierungsfilter eingesetzt, um Daten aus mehreren Formularelementen zu validieren, und zudem stellen Bereinigungsfilter sicher, dass die eingegebenen Daten sicher auf der Seite angezeigt werden können. Das Ergebnis dieses Beispiels sieht genauso aus wie das auf Seite 264.

1. Die Arrays \$user und \$error sowie die Variable \$message werden initialisiert. Dadurch können sie im Formular unten auf der Seite verwendet werden, wenn dieses zum ersten Mal geladen wird (bevor es abgeschickt wird).
2. Eine if-Anweisung überprüft, ob das Formular abgeschickt wurde.
3. Die Variable \$validation_filters enthält ein Array der Filter, die zur Validierung der Formulardaten verwendet werden.
4. filter_input_array() erfasst die Werte aus dem Formular und wendet die Validierungsfilter darauf an. Die Ergebnisse überschreiben die Werte, die in Schritt 1 in \$user gespeichert wurden. Wenn ein Wert:
 - gültig ist, wird er in dem Array gespeichert
 - ungültig ist, wird false gespeichert
 - fehlt, wird ihm der Wert null zugewiesen
5. Jeder Wert im Array \$errors wird mit einem ternären Operator gesetzt. Die Bedingung überprüft, ob das jeweilige Datenelement gültig ist. Wird der Wert gewertet als:
 - true, wird ihm eine leere Zeichenkette zugewiesen
 - false oder null, wird er eine Fehlermeldung enthalten, die dem Benutzer mitteilt, wie er die Daten zu korrigieren hat
6. Mit der PHP-Funktion implode() werden alle Werte im Array \$errors zu einem einzigen String zusammengefasst und in der Variablen \$invalid gespeichert.

7. Die Bedingung einer if-Anweisung überprüft, ob die Variable \$invalid Text enthält, der als true behandelt wird (ein leerer String wird als false behandelt).
8. Wenn die Daten ungültig sind, wird in der Variablen \$message eine Meldung gespeichert, die den Besucher auffordert, die Fehler im Formular zu korrigieren.
9. Andernfalls teilt die Variable \$message dem Benutzer mit, dass die Daten gültig waren. An dieser Stelle könnte die Seite die empfangenen Daten nun verarbeiten (und es wäre nicht nötig, das Formular erneut anzuzeigen).
10. Name und Alter, die im Array \$user gespeichert wurden, werden bereinigt, um eine sichere Anzeige auf der Seite zu gewährleisten. Dies geschieht mit der PHP-Funktion filter_var():
 - Der Name wird bereinigt, indem alle reservierten HTML-Zeichen durch Entitäten ersetzt werden.
 - Die Zahl wird so bereinigt, dass sie nur für eine Ganzzahl zulässige Zeichen enthält.
11. Der Wert in der Variablen \$message wird angezeigt.
12. Das Formular wird angezeigt. Sind die vom Benutzer eingegebenen Daten:
 - gültig, werden diese Werte in den Formularelementen angezeigt
 - ungültig, werden leere Formularelemente angezeigtWenn der Benutzer das Formular nicht abgeschickt hat, greifen die Formularelemente auf die Anfangswerte zurück, die den einzelnen Elementen des Arrays \$user in Schritt 1 zugewiesen wurden.

Bei ungültigen Daten wird eine Fehlermeldung hinter dem entsprechenden Formularelement angezeigt.

```

<?php
① $user  = ['name' => '', 'age' => '', 'terms' => '' , ];           // initialisieren
① $errors = ['name' => '', 'age' => '', 'terms' => false, ];
① $message = '';

② if ($_SERVER['REQUEST_METHOD'] == 'POST') {                                // wenn Formular versendet
    // Validierungsfilter
    $validation_filters['name']['filter']          = FILTER_VALIDATE_REGEXP;
    $validation_filters['name']['options']['regexp'] = '/^([A-z]{2,10})$/';
    $validation_filters['age']['filter']            = FILTER_VALIDATE_INT;
    $validation_filters['age']['options']['min_range'] = 16;
    $validation_filters['age']['options']['max_range'] = 65;
    $validation_filters['terms']                  = FILTER_VALIDATE_BOOLEAN;

④ $user = filter_input_array(INPUT_POST, $validation_filters); // validiere Daten

    // Fehlermeldungen erstellen
    $errors['name'] = $user['name'] ? '' : 'Name must be 2-10 letters using A-z';
    $errors['age']  = $user['age']  ? '' : 'You must be 16-65';
    $errors['terms'] = $user['terms'] ? '' : 'You must agree to the terms & conditions';
    $invalid = implode($errors);                      // Fehlermeldungen zusammenführen

    if ($invalid) {                                    // wenn Fehler aufgetreten sind
        $message = 'Please correct the following errors:'; // nicht verarbeiten
    } else {                                         // ansonsten
        $message = 'Thank you, your data was valid.'; // Daten können verarbeitet werden
    }

    // Daten bereinigen
    $user['name'] = filter_var($user['name'], FILTER_SANITIZE_FULL_SPECIAL_CHARS);
    $user['age']  = filter_var($user['age'],  FILTER_SANITIZE_NUMBER_INT);
}

?> ...

⑪ <?= $message ?>
<form action="validate-form-using-filters.php" method="POST">
    Name: <input type="text" name="name" value="<?= $user['name'] ?>">
    <span class="error"><?= $errors['name'] ?></span><br>
    Age: <input type="text" name="age" value="<?= $user['age'] ?>">
    <span class="error"><?= $errors['age'] ?></span><br>
    <input type="checkbox" name="terms" value="true"
        <?= $user['terms'] ? 'checked' : '' ??> I agree to the terms and conditions
    <span class="error"><?= $errors['terms'] ?></span><br>
    <input type="submit" value="Save">
</form>

```

ZUSAMMENFASSUNG

DATEN VON BROWSERN ERHALTEN

- Daten, die über Query-Strings und Formulare gesendet werden, fließen in die superglobalen Arrays `$_GET` und `$_POST` ein, die alle empfangenen Daten als Strings speichern.
- Wenn einen Wert in einem superglobalen Array fehlen könnte, überprüfen Sie mit der Funktion `isset()`, ob er vorhanden ist, oder geben mit dem Null-Koaleszenz-Operator einen Standardwert vor.
- Die Daten können auch mit den Funktionen `filter_input()` oder `filter_input_array()` ausgelesen werden.
- Überprüfen Sie die Daten vor der Verarbeitung. Prüfen Sie, ob die erforderlichen Daten geliefert wurden und ob sie im richtigen Format vorliegen.
- Bereinigen Sie Benutzerdaten, bevor Sie sie anzeigen, um XSS-Angriffe zu verhindern. Ersetzen Sie reservierte HTML-Zeichen durch Entitäten.
- Validierungsfilter werden in Filterfunktionen eingesetzt, um Werte zu überprüfen und in den richtigen Datentyp umzuwandeln.
- Bereinigungsfilter werden in Filterfunktionen eingesetzt, um unwünschte Zeichen zu ersetzen oder zu entfernen.



BILDER & DATEIEN

In diesem Kapitel lernen Sie, wie Sie den Besuchern Ihrer Website erlauben, Bilder auf den Server hochzuladen, und wie Sie diese sicher auf Ihren PHP-Seiten anzeigen können. Diese Techniken funktionieren auch für andere Dateitypen.

Zunächst lernen Sie, wie Anwender Bilder hochladen und wie der Server sie empfängt. Sie erfahren, dass:

- das Steuerelement zum Datei-Upload in einem HTML-Formular verwendet wird, damit Benutzer Dateien hochladen können.
- der PHP-Interpreter Daten über die Datei in das superglobale Array `$_FILES` einfügt.
- die Datei in einem temporären Ordner auf dem Server abgelegt wird.
- die Datei in einen Ordner verschoben werden muss, in dem die hochgeladenen Dateien gespeichert werden sollen.

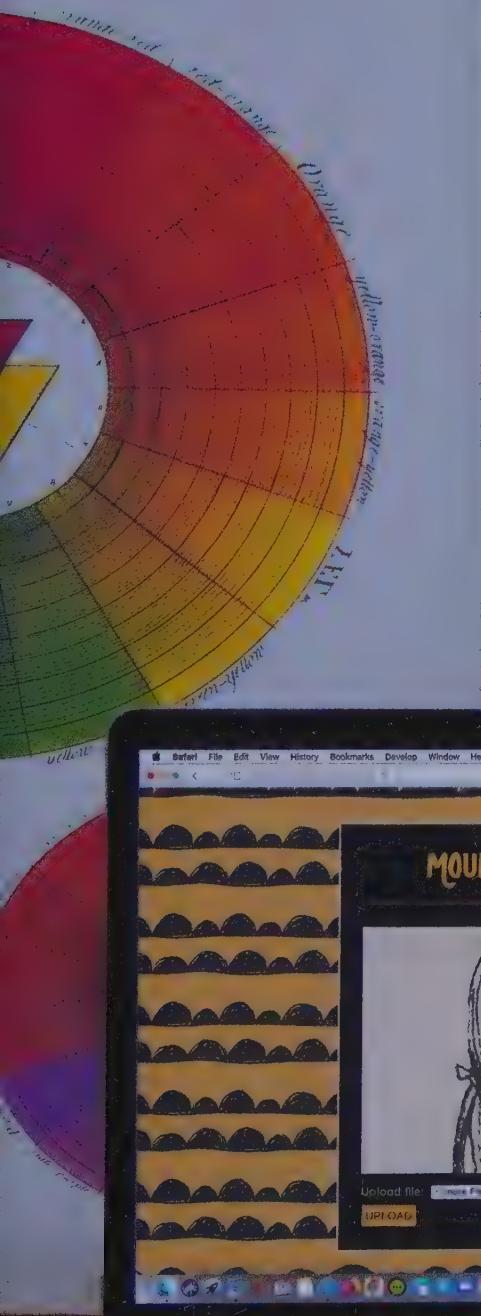
Als Nächstes lernen Sie, wie Sie die hochgeladenen Dateien validieren und überprüfen können:

- Der Dateiname darf nur zulässige Zeichen enthalten.
- Es existiert bisher noch keine Datei mit diesem Namen.
- Es handelt sich um einen zulässigen Medientyp und eine zulässige Dateierweiterung.
- Die Datei ist nicht zu groß.

Schließlich lernen Sie, wie Sie die Bilder bearbeiten und Folgendes erstellen:

- Miniaturansichten des Bilds
- zugeschnittene Bildversionen

Im Laufe des Kapitels lernen Sie noch weitere integrierte Funktionen kennen, die Sie bei diesen Aufgaben unterstützen. In diesem Kapitel demonstrieren wir diese Techniken anhand von Bildern, sie eignen sich aber auch für das Hochladen von Audio-, Video- und PDF-Dateien und weiterer Dateitypen.



DATEIEN AUS EINEM BROWSER HOCHladen

HTML-Formulare können ein Steuerelement zum Datei-Upload enthalten, mit dem Besucher Dateien auf den Server übertragen können.

Beim Erstellen eines HTML-Formulars, das Besuchern den Datei-Upload ermöglicht, muss das öffnende <form>-Tag die folgenden drei Attribute aufweisen:

- method mit dem Wert POST, um anzugeben, dass das Formular über HTTP POST gesendet werden soll (da Dateien nicht über HTTP GET gesendet werden sollten)
- enctype mit dem Wert multipart/form-data, um die Kodierungsart anzugeben, die der Browser zum Senden der Daten verwenden soll
- action, wobei der Wert der PHP-Datei entspricht, an die die Formulardaten gesendet werden sollen

Das Steuerelement zum Datei-Upload wird mit dem HTML-Element <input> erstellt. Sein type-Attribut muss den Wert file haben. Im Browser wird dann eine Schaltfläche erstellt, die ein neues Fenster zur Auswahl der hochzuladenden Datei öffnet:

```
<input type="file" name="image">
```

Wie die anderen Formularelemente sendet auch das Dateieingabe-Steuerelement ein Name/Wert-Paar an den Server:

- Der Name entspricht dem Wert des Namensattributs für dieses Dateisteuerelement (oben heißt es image)
- Der Wert ist die vom Nutzer gesendete Datei

Um die möglichen Dateitypen einzuschränken, hat das Dateieingabe-Steuerelement ein accept-Attribut. Dessen Wert sollte eine durch Kommas getrennte Liste von Medientypen sein, die von der Seite akzeptiert werden. (Medientypen werden oft als MIME-Typen bezeichnet; mehr dazu erfahren Sie hier: <http://notes.re/media-types>).

```
<input type="file" name="image"  
       accept="image/jpeg, image/png">
```

Wenn das accept-Attribut zum Einsatz kommt und die Schaltfläche zum Hochladen einer Datei betätigt wird, sperren moderne Browser alle Dateien, die nicht in der Liste der akzeptierten Typen enthalten sind, sodass sie nicht ausgewählt werden können.

Das verbessert zwar die Benutzerfreundlichkeit, aber Sie können sich nicht darauf verlassen, dass das accept-Attribut die Art der von den Besuchern hochgeladenen Dateien einschränkt, da diese die Einstellung außer Kraft setzen können und ältere Browser sie nicht unterstützen. (Chrome 10, Internet Explorer 10, Firefox 10 und Safari 6 unterstützen als erste Versionen der gängigsten Browser diese Funktion.) Daher sollten Sie den Medientyp möglichst auch auf dem Server mit PHP validieren (siehe Seite 295).

Um alle Untertypen eines Medientyps zuzulassen, können Sie anstelle eines Untertyps ein Sternchenzeichen einfügen.

Im Folgenden werden alle Bildformate zugelassen (einschließlich BMP, GIF, JPEG, PNG, TIFF und WebP):

```
<input type="file" accept="image/*">
```

1. Mit dem unten stehenden Formular können Bilder hochgeladen werden. Es wird in allen Beispielen in diesem Kapitel verwendet.

Der einleitende `<form>`-Tag benötigt ein:

- method-Attribut mit dem Wert POST
- enctype-Attribut mit dem Wert `multipart/form-data`
- action-Attribut, das auf die Datei verweist, an die die Formulardaten gesendet werden sollen (dieser Wert ändert sich in jedem Beispiel)

2. Um ein Steuerelement zum Datei-Upload zu erstellen, bekommt das `<input>`-Element ein type-Attribut mit dem Wert `file`.

Da sich die Beispiele in diesem Kapitel auf den Bild-Upload beziehen, ist der Wert des name-Attributs `image`.

3. Die submit-Schaltfläche dient zum Absenden des Formulars.

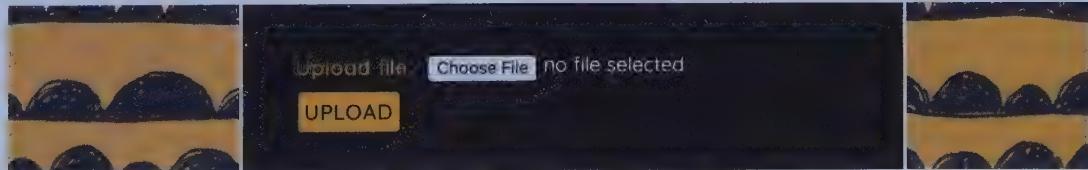
HTML

```
① <form method="post" action="filename.php" enctype="multipart/form-data">
    <label for="image"><b>Upload file:</b></label>
    ② <input type="file" name="image" accept="image/*" id="image"><br>
    ③ <input type="submit" value="Upload">
</form>
```

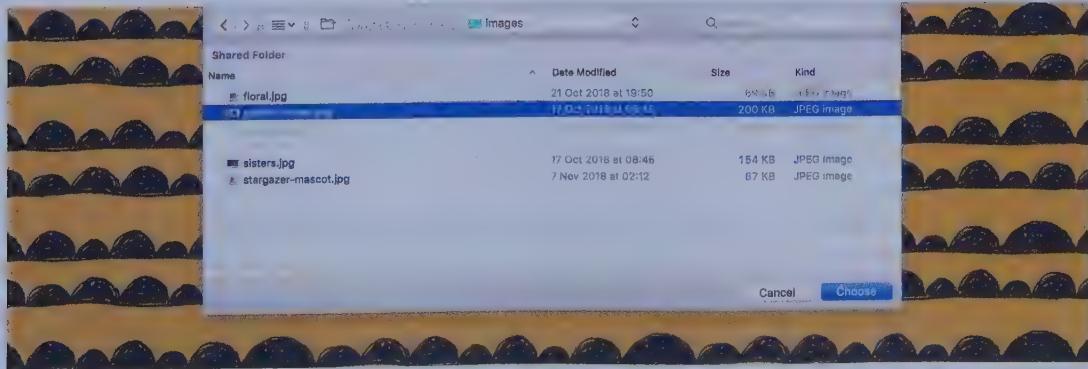
Das erste Ergebnisfeld unten zeigt das Formular, das das Upload-Steuerelement erstellt. Bei Auswahl eines Bilds wird der Text neben der Schaltfläche durch den Dateinamen ersetzt.

Im zweiten Ergebnisfeld sehen Sie das Fenster, das sich beim Anklicken der Schaltfläche Choose File öffnet. Eine Text- und eine Zip-Datei sind ausgegraut, da es sich nicht um Bilder handelt.

ERGEBNIS



ERGEBNIS



Das Erscheinungsbild des eingeblendeten Dateiauswahlfensters ist von Browser zu Browser und von Betriebssystem zu Betriebssystem unterschiedlich (Sie können es nicht mit CSS beeinflussen).

DATEIEN AUF DEM SERVER EMPFANGEN

Wird eine Datei über eine Webseite hochgeladen, speichert der Webserver sie in einem temporären Ordner, und der PHP-Interpreter verwaltet im superglobalen Array `$_FILES` Einzelheiten über die Datei.

Ein Formular kann mehrere Steuerelemente zum Datei-Upload enthalten, daher erstellt der PHP-Interpreter für jedes vom Formular übermittelte Upload-Steuerelement ein Element im superglobalen Array `$_FILES`.

Der Name des Elements entspricht dem Namen des Datei-Upload-Steuerelements, und sein Wert ist ein Array mit Informationen über die Datei, die über dieses Formularelement hochgeladen wurde.

Die folgende Tabelle zeigt die im superglobalen Array `$_FILES` für jede hochgeladene Datei abgelegten Informationen.

Die Bilder in diesem Kapitel werden mit einem Datei-Upload-Steuerelement mit dem Namen `image` hochgeladen. Das Array `$_FILES` enthält also das Element `image`, und dessen Wert ist ein Array mit Informationen über das jeweilige Bild.

SCHLÜSSEL	WERT	SO GREIFEN SIE AUF DEN WERT ZU
<code>name</code>	Dateiname	<code>\$_FILES['image']['name']</code>
<code>tmp_name</code>	temporärer Speicherort der Datei (vom PHP-Interpreter festgelegt)	<code>\$_FILES['image']['tmp_name']</code>
<code>size</code>	Größe in Bytes	<code>\$_FILES['image']['size']</code>
<code>type</code>	Medientyp (je nach Browser)	<code>\$_FILES['image']['type']</code>
<code>error</code>	0, wenn die Datei erfolgreich hochgeladen wurde, oder ein Fehlercode, falls es ein Problem gab	<code>\$_FILES['image']['error']</code>

Nach dem Upload einer Datei sollte der PHP-Code überprüfen, ob der PHP-Interpreter beim Hochladen keine Fehler festgestellt hat.

Wenn der Schlüssel `error` im für diese Datei erstellten Array den Wert 0 hat, dann bedeutet dies, dass der PHP-Interpreter keine Fehler gefunden hat.

```
if ($_FILES['image']['errors'] == 0) {  
    // Process image  
} else {  
    // Show error message  
}
```

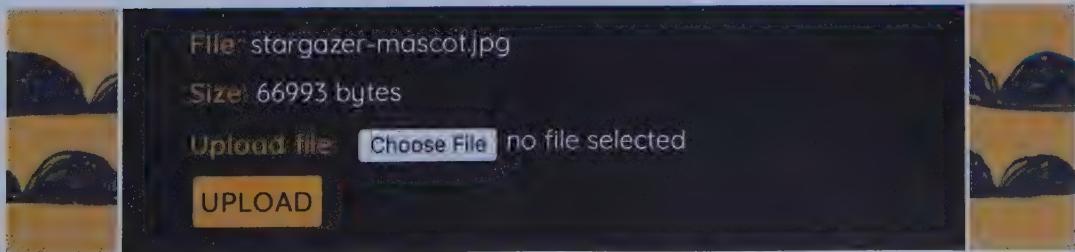
ÜBERPRÜFEN, OB EINE DATEI HOCHGELADEN WURDE

1. Die Variable \$message wird mit einer leeren Zeichenkette initialisiert. Sobald das Formular abgeschickt wurde, wird sie mit einer Nachricht befüllt.
2. Wenn das Formular über HTTP POST versendet wurde ...
3. Eine if-Anweisung überprüft, dass keine Fehler vorliegen.
4. Liegen keine Fehler vor, werden der Name und die Größe der Datei in \$message gespeichert.
5. Andernfalls wird in \$message eine Fehlermeldung abgelegt.
6. Der Wert in der Variablen \$message wird angezeigt.

PHP

```
<?php  
① $message = '';  
② if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
③     if ($_FILES['image']['error'] == 0) {  
④         $message = '<b>File:</b> ' . $_FILES['image']['name'] . '<br>'; // Dateiname  
        $message .= '<b>Size:</b> ' . $_FILES['image']['size'] . ' bytes'; // Dateigröße  
    } else {  
        $message = 'The file could not be uploaded.'; // Sonst  
    }  
}  
?  
⑥ <?= $message ?>  
<form method="POST" action="upload-file.php" enctype="multipart/form-data">  
    <label for="image"><b>Upload file:</b></label>  
    <input type="file" name="image" accept="image/*" id="image"><br>  
    <input type="submit" value="Upload">  
</form>
```

ERGEBNIS



EINE DATEI AN IHREN ZIELORT VERSCHIEBEN

Die PHP-Funktion `move_uploaded_file()` verschiebt eine Datei von ihrem temporären Speicherort an den gewünschten Zielspeicherort auf dem Server.

Wenn eine Datei auf den Server hochgeladen wird, erhält sie einen temporären Dateinamen und wird in einem temporären Ordner gespeichert. (Der temporäre Dateiname wird vom PHP-Interpreter generiert.)

Der PHP-Interpreter löscht die temporäre Datei nach Ausführung des Skripts aus diesem Ordner. Um eine hochgeladene Datei also auf dem Server zu speichern, müssen Sie die Funktion `move_uploaded_file()` aufrufen, um sie an einen anderen Speicherort zu verschieben. Diese hat zwei Parameter:

- den temporären Speicherort der Datei
- den Zielspeicherort für die Datei

Sie liefert `true` zurück, wenn die Datei an den neuen Speicherort verschoben werden konnte, und `false`, wenn das nicht gelingt.

Die Angabe für den Zielspeicherort setzt sich wie folgt zusammen:

- Pfad zu dem Ordner, in dem die hochgeladene Datei gespeichert werden soll (dieser Ordner muss zuerst erstellt worden sein, bevor Sie versuchen, eine Datei dorthin zu verschieben)
- Dateiname (der ursprüngliche Dateiname oder ein neuer Name)

Wenn Sie den ursprünglichen Namen der hochgeladenen Datei verwenden möchten, können Sie diesen aus dem Array abrufen, das der PHP-Interpreter für diese Datei erstellt hat. Der entsprechende Schlüssel lautet `name`. Unten wird der Zielpfad in der Variablen `$destination` gespeichert. Er setzt sich zusammen aus dem Ordner `uploads`, gefolgt vom ursprünglichen Dateinamen, der beim Hochladen des Bilds verwendet wurde.

```
NEUER ORDNER  
_____  
$destination = '../uploads/' . $_FILES['image']['name'];  
move_uploaded_file($_FILES['image']['tmp_name'], $destination);  
_____  
TEMPORÄRER SPEICHERORT           ZIEL-DATEIPFAD
```

DATEIZUGRIFFSRECHTE

Folgende Berechtigungen sollten für das Zielverzeichnis festgelegt werden:

- Dem Webserver das Lesen/Schreiben von Dateien erlauben – dies ermöglicht die Speicherung und Anzeige von Bildern.
- Ausführungsberechtigungen deaktivieren – dies verhindert die Ausführung bösartiger Skripte.

ÜBERPRÜFEN, OB EINE DATEI HOCHGELADEN WURDE

Die PHP-Funktion `move_uploaded_file()` überprüft vor dem Verschieben einer Datei, ob überhaupt eine solche per HTTP POST hochgeladen wurde.

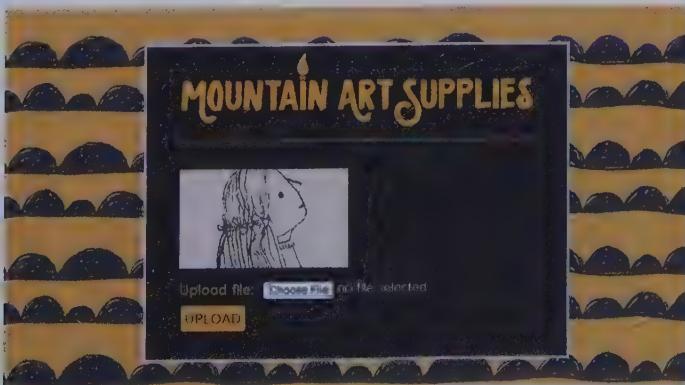
Wenn Sie eine Datei bereits vor dem Verschieben verwenden müssen, können Sie diese Überprüfung auch mit der PHP-Funktion `is_uploaded_file()` durchführen. (Dies kann helfen zu verhindern, dass jemand auf andere Dateien zugreift.)

EINE HOCHGELADENE DATEI VERSCHIEBEN

PHP

```
<?php  
$message = '';  
① $moved = false; // initialisieren  
② if ($_SERVER['REQUEST_METHOD'] == 'POST') { // wenn gesendet  
    if ($_FILES['image']['error'] == 0) { // + keine Fehler  
        // temporärer Pfad und neuer Zielpfad  
        $temp = $_FILES['image']['tmp_name'];  
        $path = 'uploads/' . $_FILES['image']['name'];  
        // verschiebe Datei und speichere Ergebnis in $moved  
        $moved = move_uploaded_file($temp, $path);  
    }  
    if ($moved == true) { // wenn Verschieben erfolgreich, Bild  
        $message = '';  
    } else { // sonst Fehlermeldung  
        $message = 'The file could not be saved.';  
    }  
}  
?> ...  
⑨ <?= $message ?>
```

ERGEBNIS



1. Die Variable `$moved` wird mit dem Wert `false` initialisiert.

Dieser Wert ändert sich in `true`, wenn das Bild erfolgreich verschoben wurde.

2. Wenn das Formular abgeschickt wurde und keine Fehler aufgetreten sind ...

3. `$temp` enthält den vom PHP-Interpreter gewählten temporären Dateispeicherort.

4. `$path` enthält den Zielspeicherort für die Datei. (Der Dateiname beim Hochladen wird beibehalten.)

5. `move_uploaded_file()` versucht, die Datei von ihrem temporären Speicherort (in `$temp`) an den neuen Speicherort (in `$path`) zu verschieben. Die Funktion liefert `true` zurück, wenn es geklappt hat, oder `false`, wenn es fehlgeschlagen ist. Dieser Wert ersetzt den in Schritt 1 in der Variablen `$moved` abgelegten Wert.

6. Eine bedingte Anweisung testet, ob `$moved` den Wert `true` hat (in diesem Fall hat das Verschieben funktioniert).

7. Trifft dies zu, so wird in `$message` ein ``-HTML-Tag zur Anzeige des hochgeladenen Bilds gespeichert.

8. Andernfalls wird in `$message` eine Fehlermeldung abgelegt.

9. Der in `$message` gespeicherte Wert wird auf der Seite angezeigt.

DATEINAMEN UND DOPPELTE DATEIEN BEREINIGEN

Bevor Sie eine Datei von ihrem temporären Speicherort verschieben, sollten Sie:

- möglichsterweise problematische Zeichen aus dem Dateinamen entfernen und
- sicherstellen, dass keine andere Datei mit demselben Namen überschrieben wird.

Zeichen wie kaufmännische Und-Zeichen, Doppelpunkte, Punkte und Leerzeichen sollten Sie aus den Dateinamen entfernen, da sie Probleme verursachen können. Zu diesem Zweck können Sie alle Zeichen außer A-Z, a-z und 0-9 durch einen Bindestrich ersetzen.

- Benutzen Sie die PHP-Funktion pathinfo() (siehe Seite 228), um den Dateinamen (basename) und dessen Erweiterung (extension) zu ermitteln.

```
① $basename = pathinfo($filename, PATHINFO_FILENAME);
② $extension = pathinfo($filename, PATHINFO_EXTENSION);
③ $filepath = 'uploads/' . $basename . '.' . $extension;
```

Wenn beim Aufruf von move_uploaded_file() bereits eine Datei mit demselben Namen existiert, wird die alte Datei durch die neue ersetzt. Um dies zu verhindern, muss jede Datei einen eindeutigen Namen haben:

- Setzen Sie einen Zähler auf 1 und speichern Sie ihn in der Variablen \$i.
- Überprüfen Sie in der Bedingung einer while-Schleife anhand der PHP-Funktion file_exists() (siehe Seite 228), ob bereits eine Datei mit demselben Namen existiert.

```
④ $i = 1;
⑤ while (file_exists('uploads/' . $filename)) {
⑥     $i = $i + 1;
⑦     $filename = $basename . $i . '.' . $extension;
}
```

- Mit der PHP-Funktion preg_replace() (siehe Seite 214) ersetzen Sie alle Zeichen im Dateinamen außer A-Z, a-z und 0-9 durch einen Bindestrich.

- Erstellen Sie den Zielpfad, indem Sie das Upload-Verzeichnis, den Dateinamen, einen Punkt und die Dateinamenerweiterung zusammenfügen. Dieser Wert sollte in einer Variablen gespeichert werden.

- Ist dies der Fall, erhöhen Sie den Zählerwert um 1.

- Aktualisieren Sie den Dateinamen, indem Sie den Wert im Zähler nach dem Stammnamen und vor der Erweiterung hinzufügen. Gibt es z.B. bereits upload.jpg, nennen Sie die neue Datei upload1.jpg.

Die Bedingung der Schleife wird dann erneut ausgeführt, um zu überprüfen, ob der neue Dateiname existiert. Die Schleife wiederholt die Schritte 5 - 7, bis die Datei einen eindeutigen Namen erhält.

DATEIGRÖSSE UND -TYP ÜBERPRÜFEN

Um sicherzustellen, dass eine Website eine hochgeladene Datei verarbeiten kann, sollten Sie sie vor dem Verschieben Folgendes überprüfen:

- a) dass die Datei nicht zu groß ist (große Dateien brauchen länger zum Herunterladen/Verarbeiten) und
- b) dass die Website mit dem Medientyp und der Dateinamenerweiterung umgehen kann.

Sie können in `php.ini` oder `.htaccess` eine maximale Dateigröße für den Upload festlegen (siehe die Seiten 196 bis 199), oder Sie schreiben einen eigenen Validierungs-Code, um die maximale Dateigröße für Uploads auf Ihren Seiten zu beschränken.

Um zu sehen, ob eine Datei größer ist als die in `php.ini` oder `.htaccess` festgelegte maximale Upload-Größe, schauen Sie in das Array `$_FILES`. Wenn ja, enthält der Schlüssel `error` für diese Datei den Wert 1.

```
$error = ($_FILES['image']['error']) == 1 ? 'Too Large' : '';
$error = ($_FILES['image']['size'] <= 5242880) ? '' : 'Too Large';
```

Durch die Validierung des Medientyps und der Dateinamenerweiterung einer Datei kann gewährleistet werden, dass eine Website eine Datei problemlos verarbeiten kann. Unten:

1. `$allowed_types` ist ein Array der erlaubten Medientypen.
2. Die PHP-Funktion `mime_content_type()` versucht, den Medientyp der Datei zu ermitteln, und speichert ihn in `$type`.
3. Die PHP-Funktion `in_array()` überprüft, ob der Medientyp dieser Datei im Array der erlaubten Medientypen enthalten ist.

```
① $allowed_types = ['image/jpeg', 'image/png', 'image/gif',];
② $type = mime_content_type($_FILES['image']['tmp_name']);
③ $error = in_array($type, $allowed_types) ? '' : 'Wrong file type';
④ $allowed_exts = ['jpeg', 'jpg', 'png', 'gif',];
⑤ $filename = strtolower($_FILES['image']['name']);
⑥ $ext = pathinfo($filename, PATHINFO_EXTENSION);
⑦ $error .= in_array($ext, $allowed_exts) ? '' : 'Wrong extension';
```

Sie können die Größe einer Datei auch im Array `$_FILES` überprüfen; der Schlüssel `size` enthält die Größe dieser Datei in Bytes. Die beiden nachfolgenden ternären Operatoren dienen der Durchführung dieser beiden Überprüfungen. Die Bedingung des:

- ersten ternären Operators überprüft, ob der Fehlercode 1 lautet,
- die des zweiten überprüft, ob die Größe über 5 MB liegt.

4. `$allowed_exts` enthält ein Array der erlaubten Erweiterungen.
5. Der Dateiname wird in Kleinbuchstaben umgewandelt und in `$filename` gespeichert.
6. Die Dateinamenerweiterung wird abgefragt und in `$ext` gespeichert.
7. Die PHP-Funktion `in_array()` prüft, ob diese Dateinamenerweiterung im Array der zulässigen Erweiterungen enthalten ist.

DATEI-UPLOADS VALIDIEREN

Dieses Beispiel enthält den Code zum Hochladen, Validieren und Speichern einer Datei.

1. Es werden sechs Variablen erstellt, und sie enthalten:

- Ergebnis, ob die Datei hochgeladen wurde oder nicht
- Erfolgs-/Fehlermeldung, die der Benutzer sieht
- Fehler, wenn es Probleme mit dem Bild gibt
- Pfad zu dem Ordner, in dem die hochgeladenen Dateien gespeichert sind
- Maximale Dateigröße in Bytes
- Erlaubte Medientypen
- Erlaubte Dateinamenerweiterungen

2. Die Funktion `create_filename()` wird definiert.

Sie nutzt den Code von Seite 294, um den Dateinamen zu bereinigen und sicherzustellen, dass der Dateiname eindeutig ist, und liefert dann den neuen Dateinamen zurück. Ihre beiden Parameter sind der:

- Dateiname
- relative Pfad zum Ordner, in dem die Datei gespeichert werden soll

3. Eine `if`-Anweisung überprüft, ob das Formular abgeschickt wurde.

4. Mithilfe eines ternären Operators wird festgestellt, ob beim Hochladen des Bilds ein Fehler aufgetreten ist, weil das Bild größer als die in der `php.ini` oder `.htaccess` festgelegte maximale Größe war. Falls ja, wird eine Fehlermeldung in `$error` gespeichert.

5. Eine weitere `if`-Anweisung überprüft, ob die Datei fehlerfrei hochgeladen wurde.

6. Die Dateigröße wird überprüft. Wenn sie kleiner oder gleich der in Schritt 1 in `$max_size` gespeicherten Maximalgröße ist, enthält `$error` eine leere Zeichenkette; ist sie größer als die maximal zulässige Größe, enthält `$error` die Meldung 'too big'.

7. Die PHP-Funktion `mime_content_type()` ermittelt den Medientyp der Datei und speichert ihn in `$type`.

8. Die PHP-Funktion `in_array()` prüft, ob der in `$type` gespeicherte Medientyp im Array `$allowed_types` enthalten ist. Trifft dies zu, wird in der Variablen `$error` eine leere Zeichenkette abgelegt. Ist dies nicht der Fall, wird eine Fehlermeldung in `$error` gespeichert.

9. Die PHP-Funktion `pathinfo()` ermittelt die Dateinamenerweiterung des hochgeladenen Bilds. Diese Funktion wird innerhalb der PHP-Funktion `strtolower()` aufgerufen, um sicherzustellen, dass die Erweiterung aus Kleinbuchstaben besteht. Sie wird dann in `$ext` gespeichert.

10. Die PHP-Funktion `in_array()` überprüft, ob die Dateinamenerweiterung zulässig ist. Ist dies der Fall, wird eine leere Zeichenkette in der Variablen `$error` abgelegt. Trifft dies nicht zu, wird eine Meldung gespeichert, die besagt, dass es sich um die falsche Erweiterung handelt.

11. Die Bedingung einer `if`-Anweisung überprüft, ob `$error` einen Wert enthält, der nicht als `true` betrachtet wird. Eine leere Zeichenkette wird als `false` gewertet (es liegen also keine Fehler vor).

12. Wenn keine Fehler vorliegen, wird die Funktion `create_filename()` (aus Schritt 2) aufgerufen, um sicherzustellen, dass der Dateiname bereinigt und eindeutig ist.

13. `$destination` enthält den Pfad zum Speichern der neuen Datei.

14. Die PHP-Funktion `move_uploaded_file()` wird aufgerufen, um die Datei von ihrem temporären Speicherort in den Ordner `uploads` zu verschieben. Sie liefert `true` zurück, wenn dies funktioniert, und `false`, wenn nicht. Das Ergebnis wird in der Variablen `$moved` gespeichert.

15. Wenn die Variable `$moved` den Wert `true` hat, wurde das Bild hochgeladen, hat alle Prüfungen bestanden und wurde gespeichert, sodass in der Variablen `$message` ein ``-HTML-Tag zur Anzeige des Bilds abgelegt wird.

16. Wenn nicht, wird eine Fehlermeldung in `$message` gespeichert.

17. Der in der Variablen `$message` gespeicherte Wert wird vor dem Upload-Formular angezeigt.

```

<?php
① $moved      = false;                                // initialisieren
$message    = '';
$error       = '';
$upload_path = 'uploads/';                            // Upload-Pfad
$max_size    = 5242880;                               // maximale Dateigröße (in Bytes)
$allowed_types = ['image/jpeg', 'image/png', 'image/gif',]; // erlaubte Dateitypen
$allowed_exts = ['jpeg', 'jpg', 'png', 'gif',];        // erlaubte Dateinamenerweiterungen

function create_filename($filename, $upload_path)          // Funktion zur Dateinamenerstellung
{
    $basename   = pathinfo($filename, PATHINFO_FILENAME); // Stammname abrufen
    $extension  = pathinfo($filename, PATHINFO_EXTENSION); // Erweiterung abrufen
    $basename   = preg_replace('/[^A-z0-9]/', '-', $basename); // Stammname bereinigen
    $i          = 0;                                     // Zähler
    ② while (file_exists($upload_path . $filename)) {    // Wenn Datei existiert
        $i          = $i + 1;                           // Zähler aktualisieren
        $filename  = $basename . $i . '.' . $extension; // Neuer Dateipfad
    }
    return $filename;                                  // Dateinamen zurückgeben
}

③ if ($_SERVER['REQUEST_METHOD'] == 'POST') {           // wenn Formular übermittelt
    ④ $error = ($_FILES['image']['error'] == 1) ? 'too big' : ''; // Größenfehler überprüfen

    ⑤ if ($_FILES['image']['error'] == 0) {               // wenn keine Upload-Fehler
        $error .= ($_FILES['image']['size'] <= $max_size) ? '' : 'too big'; // Größe prüfen
        // prüfen, ob Medientyp im Array $allowed_types enthalten ist
        ⑦ $type   = mime_content_type($_FILES['image']['tmp_name']);
        ⑧ $error .= in_array($type, $allowed_types) ? '' : 'wrong type';
        // prüfen, ob Dateinamenerweiterung im Array $allowed_exts enthalten ist
        ⑨ $ext    = strtolower(pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION));
        ⑩ $error .= in_array($ext, $allowed_exts) ? '' : 'wrong file extension';
        // wenn keine Fehler, neuen Dateipfad anlegen und versuchen, Datei zu verschieben
        ⑪ if (!$error) {
            ⑫ $filename  = create_filename($_FILES['image']['name'], $upload_path);
            ⑬ $destination = $upload_path . $filename;
            ⑭ $moved     = move_uploaded_file($_FILES['image']['tmp_name'], $destination);
        }
    }

    ⑮ if ($moved == true) {                             // wenn Datei verschoben wurde
        $message = 'Uploaded:<br>'; // Bild zeigen
    } else {
        $message = '<b>Could not upload file:</b> ' . $error; // Fehler zeigen
    }
}

⑯ ?> ... <?= $message ?> <!-- Formular zeigen-->

```

BILDER SKALIEREN

Hochgeladene Bilder werden von Websites oft auf eine einheitliche Größe gebracht. Damit sieht die Seite übersichtlicher aus und wird schneller geladen. Um die Größe eines Bilds zu ändern, benötigen Sie sein Seitenverhältnis: Breite geteilt durch Höhe.

Hochgeladene Bilder werden aus zwei Gründen oftmals in der Größe angepasst:

- Eine Reihe von Bildern in ähnlicher Größe sieht schöner aus als mehrere Bilder in sehr unterschiedlichen Größen.
- Ist eine hochgeladene Datei größer als die Darstellungsgroße, verlängert sich die Ladezeit der Seite.

Beim Skalieren von Bildern sollten Sie das Seitenverhältnis (Breite zu Höhe) beibehalten, da die Bilder sonst verzerrt aussehen (so wie rechts gezeigt).

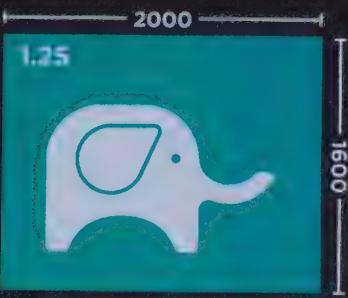
Wenn alle Bilder genau die gleiche Größe haben sollen, können Sie das Bild zuschneiden (nur einen Teil davon auswählen) und dann die Größe dieser Auswahl unter Beibehaltung des Seitenverhältnisses ändern (siehe die Seiten 300 bis 301).

QUERFORMAT

Querformatige Bilder sind breiter als hoch, sodass sich ein Seitenverhältnis größer als 1 ergibt.

Im unteren Beispiel beträgt die Breite 2000 und die Höhe 1600, das Seitenverhältnis ist also:

$$2000 \div 1600 = 1.25$$



QUADRAT

Quadratische Bilder sind genauso hoch wie breit, daher beträgt das Seitenverhältnis genau 1.

Im unteren Beispiel beträgt die Breite 2000 und die Höhe 2000, das Seitenverhältnis ist also:

$$2000 \div 2000 = 1$$



HOCHFORMAT

Hochformatige Bilder sind stets höher als breit, sodass sich ein Seitenverhältnis kleiner als 1 ergibt.

Im unteren Beispiel beträgt die Breite 1600 und die Höhe 2000, das Seitenverhältnis ist also:

$$1600 \div 2000 = 0.8$$



Unten sehen Sie, wie Sie bei der Größenänderung die neue Breite und Höhe eines Bilds berechnen können. Wenn Sie das Seitenverhältnis des Originalbilds beibehalten, erscheint das Bild in der neuen Größe nicht verzerrt.

Breite und Höhe des Rahmens müssen festgelegt werden. Dies entspricht der maximalen Breite und Höhe, die das Bild in der neuen Größe haben kann. In diesem Beispiel liegt die maximale Breite und Höhe bei 1000.

1

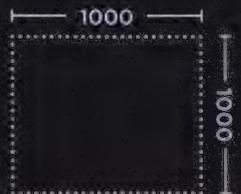
Ermitteln Sie die Breite und Höhe des hochgeladenen Originalbilds.

Berechnen Sie damit sein Seitenverhältnis (Breite : Höhe).

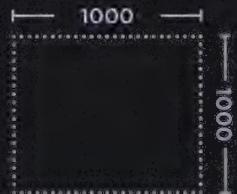
Damit eine Reihe skalierter Bilder einheitlicher aussieht, wird ihre Größe so geändert, dass sie in einen quadratischen Begrenzungsrahmen passen, der die maximale Breite und Höhe des Bilds festlegt.

Beim Ändern der Bildgröße füllt die längere Bildseite (Breite oder Höhe) den Rahmen, und die kürzere Seite wird anhand des Seitenverhältnisses berechnet.

QUERFORMAT



HOCHFORMAT



2

Ist die Breite größer als die Höhe, liegt das Bild im Querformat vor, ansonsten im Hochformat.

Gleichen Sie die längere Bildseite an die Größe des Rahmens an.

SEITENVERHÄLTNIS: 1.25

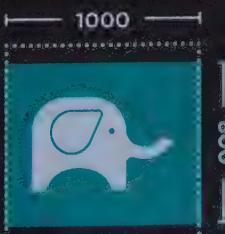


SEITENVERHÄLTNIS: 0.8

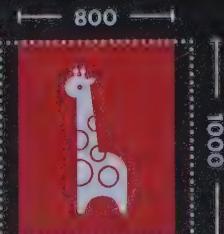


3

Berechnen Sie die Länge der kürzeren Seite des skalierten Bilds. Querformat: Teilen Sie die Rahmenhöhe durch das Seitenverhältnis. Hochformat: Multiplizieren Sie die Rahmenbreite mit dem Seitenverhältnis.



Ein querformatiges Bild füllt den Rahmen nicht in seiner ganzen Höhe aus.



Ein hochformatiges Bild füllt den Rahmen nicht in seiner ganzen Breite aus.

BILDER ZUSCHNEIDEN

Durch das Zuschneiden von Bildern können Sie mehrere genau gleich große Bilder erstellen, die dann genau in die dafür vorgesehenen Rahmen passen. Beim Zuschneiden wird ein Teil des Originalbilds entfernt.

Beim Zuschneiden müssen Sie den Teil des Originalbilds auswählen, den Sie beibehalten möchten.

Um mehrere Bilder in eine einheitliche Form zu bringen, sollte der Zuschnittsbereich jedes Bilds das gleiche Seitenverhältnis haben.

Sobald Sie einen Bereich ausgewählt haben, können Sie dessen Größe ändern, um sicherzustellen, dass alle Bilder die gleiche Größe erhalten.

Zur Auswahl des gewünschten Bildausschnitts benötigen Sie vier Werte:

- **Auswahlbreite:** Die Breite des beizubehaltenden Bildbereichs, ausgehend vom X-Versatz
- **Auswahlhöhe:** Die Höhe des beizubehaltenden Bildbereichs, ausgehend vom Y-Versatz
- **X-Versatz:** Der Abstand von der linken Bildkante bis zu dem Punkt, an dem die Auswahl beginnen soll
- **Y-Versatz:** Der Abstand von der oberen Bildkante bis zu dem Punkt, an dem die Auswahl beginnen soll

Es gibt JavaScript-Tools, mit denen Bilder bereits vor dem Upload im Browser zugeschnitten werden können. Einige davon finden Sie hier:
<http://notes.re/php/images/crop-javascript>.

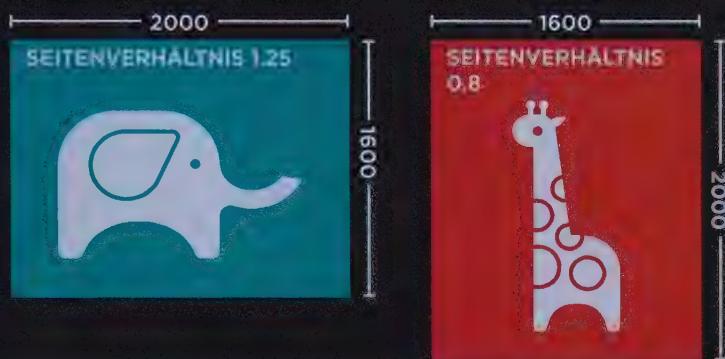


Damit die hochgeladenen Bilder hinterher alle die gleiche Größe haben, müssen Sie die gewünschte Breite und Höhe angeben. Diese Werte werden zur Berechnung des neuen Seitenverhältnisses (Breite ÷ Höhe) verwendet.



1

Ermitteln Sie die Breite und Höhe des hochgeladenen Bilds und berechnen Sie dessen Seitenverhältnis (Breite : Höhe).

**2**

Wählen Sie den relevanten Bildbereich aus, der erhalten bleiben soll.

Diese Auswahl sollte das gleiche Seitenverhältnis haben wie das neue Bild.

Die Berechnungen für die Auswahl und den Versatz sind unten dargestellt.

X-VERSATZ **AUSWAHLBREITE**

200 1600



AUSWAHLBREITE

1600



Y-VERSATZ
200

AUSWAHLHÖHE
1600

Wenn das neue Seitenverhältnis kleiner ist als das des hochgeladenen Bilds:

- Auswahlbreite = Originalhöhe x neues Seitenverhältnis
- Auswahlhöhe = ursprüngliche Höhe
- X-Versatz = (Originalbreite - Auswahlbreite) / 2
- Y-Versatz = 0

Andernfalls verwenden Sie diese Berechnungen:

- Auswahlbreite = ursprüngliche Breite
- Auswahlhöhe = ursprüngliche Breite x neues Seitenverhältnis
- X-Versatz = 0
- Y-Versatz = (Originalhöhe - Auswahlhöhe) / 2

3

Der zugeschnittene Bereich wird auf die Größe des neuen Bilds skaliert (siehe linke Seite).



BILDBEARBEITUNG MIT EXTENSIONS

Extensions erweitern den PHP-Interpreter um zusätzliche Funktionalitäten, sodass er zusätzliche Aufgaben übernehmen kann. GD und Imagick sind zwei beliebte Extensions, mit deren Hilfe der PHP-Interpreter die Größe von Bildern anpassen und diese zuschneiden kann.

Wenn Extensions auf einem Webserver installiert werden, stellen sie in der Regel zusätzliche Funktionen oder Klassen bereit, auf die Ihre PHP-Seiten zugreifen können (so wie Ihr Code auf die bereits in PHP integrierten Funktionen und Klassen zurückgreift).

Die Extensions GD und Imagick übernehmen ähnliche Aufgaben wie die Grundfunktionen von Photoshop, aber statt hierfür eine grafische Benutzeroberfläche bereitzustellen, erlauben sie Ihnen eine Bildbearbeitung mittels PHP-Code.

Im weiteren Verlauf dieses Kapitels erfahren Sie, wie Sie Bilder mit GD skalieren und mit Imagick sowohl skalieren als auch zuschneiden können. (Ein Beispiel für den Bildzuschnitt mit GD finden Sie online unter: <http://notes.re/php/gd-crop>.)

GD ist in der Anwendung komplizierter als Imagick, aber es wird seit PHP 4.3 standardmäßig mit dem PHP-Interpreter installiert, während Sie Imagick vor dessen Verwendung separat auf dem Webserver installieren müssen.

GD VERWENDEN

Wenn Sie MAMP auf einem Mac verwenden, sollte GD standardmäßig aktiviert sein.

Wenn Sie XAMPP auf einem PC verwenden, müssen Sie die GD-Extension wahrscheinlich erst aktivieren, bevor Sie sie verwenden können, siehe: <http://notes.re/php/enable-gd>.

Um die Größe und den Zuschnitt eines Bilds mit GD zu ändern, müssen Sie fünf Funktionen von GD aufrufen (siehe unten).

GD verfügt über Funktionen zum Öffnen verschiedener Medientypen (GIF, JPEG, PNG, WEBP und so weiter) und entsprechende Funktionen zum Speichern (der kursive *Medientyp* unten wird durch den Dateityp ersetzt – siehe rechte Seite).

FUNKTION	BESCHREIBUNG
<code>getimagesize()</code>	Abmessungen und Medientyp eines Bilds ermitteln
<code>imagecreatefrom<i>mediatype()</i></code>	Das Bild öffnen (<i>mediatype</i> bitte durch den Medientyp des Bilds ersetzen)
<code>imagecreatetruecolor()</code>	Ein neues leeres Bild mit den Abmessungen des skalierten oder beschnittenen Bilds erstellen
<code>imagecopyresampled()</code>	Den ausgewählten Teil des Originalbilds nehmen, skalieren und in das im letzten Schritt erstellte neue Bild einfügen
<code>image<i>mediatype()</i></code>	Das Bild speichern (<i>mediatype</i> bitte durch den Medientyp des Bilds ersetzen)

DEN MEDIENTYP BESTIMMEN

Um die richtige Funktion zum Öffnen oder Speichern eines Bilds zu wählen, müssen Sie seinen Medientyp kennen.

Die GD-Funktion `getimagesize()` benötigt einen Bildpfad als Argument. Sie liefert ein Array zurück, das Daten über das Bild einschließlich seines Medientyps enthält.

Die nebenstehende Tabelle zeigt die in diesem Array enthaltenen Daten (die Schlüssel sind eine Mischung aus Zahlen und Wörtern).

BILDER ÖFFNEN UND SPEICHERN

Der Medientyp des Bilds kann in eine `switch`-Anweisung (wie auf der nächsten Seite gezeigt) einfließen, um so die passende Funktion zum Öffnen oder Speichern des Bilds aufzurufen.

Die nebenstehende Tabelle zeigt einige der Funktionen, die GD zum Öffnen und Speichern von Bildformaten anbietet.

BILDER SKALIEREN UND BESCHNEIDEN

Die Funktion `imagecopyresampled()` kopiert einen Teil eines Bilds (oder das komplette Bild) in ein neues leeres Bild.

Hierzu verfügt die Funktion über 10 Parameter, aber es ist einfacher, sich diese als 5 Parameterpaare vorzustellen:

- `$new, $orig`
Neues und ursprüngliches Bild (vor Funktionsaufruf in Variablen gespeichert – siehe Seite 304)
- `$new_x, $new_y`
X- und Y-Versatz für die Positionierung den kopierten Bereichs im neuen Bild
- `$orig_x, $orig_y`
X- und Y-Versatz für die Auswahl aus dem Originalbild
- `$new_width, $new_height`
Breite und Höhe der Auswahl im neuen Bild
- `$orig_width, $orig_height`
Breite und Höhe der Auswahl im Originalbild

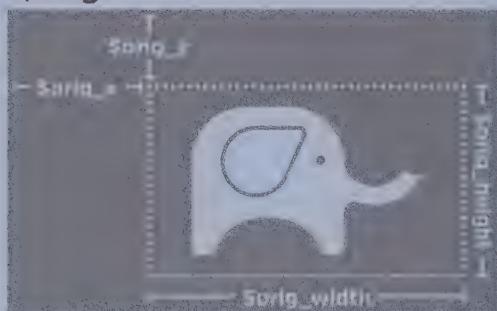
```
imagecopyresampled($new, $orig, $new_x, $new_y, $orig_x, $orig_y,  
$new_width, $new_height, $orig_width, $orig_height);
```

SCHLÜSSEL BESCHREIBUNG

0	Bildbreite (in Pixeln)
1	Bildhöhe (in Pixeln)
2	Konstante zur Beschreibung des Bildtyps
3	String mit Abmessungen zur Verwendung in einem <code></code> -Tag: <code>height="yyy" width="xxx"</code>
mime	Medientyp des Bilds
chan-	3, wenn es sich um ein RGB-Bild handelt, nels 4 für CMYK
bits	Anzahl der verwendeten Bits je Farbe

FORMAT	ÖFFNEN	SPEICHERN
GIF	<code>imagecreatefrom-</code> <code>gif()</code>	<code>image-</code> <code>gif()</code>
JPEG	<code>imagecreatefrom-</code> <code>jpeg()</code>	<code>image-</code> <code>jpeg()</code>
PNG	<code>imagecreatefrompng()</code>	<code>image-</code> <code>png()</code>
WEBP	<code>imagecreatefrom-</code> <code>webp()</code>	<code>image-</code> <code>webp()</code>

\$orig



\$new



BILDER MIT GD SKALIEREN

Die Funktion auf der rechten Seite nutzt GD zur Erstellung eines Miniaturbilds. Das Seitenverhältnis der Miniaturansicht bleibt gegenüber dem Original unverändert. Die neue Größe ergibt sich aus der maximalen Breite und Höhe, die als Argumente angegeben werden.

Das vollständige Beispiel im Code-Download folgt dem Beispiel auf Seite 297. Der einzige Unterschied besteht darin, dass ein Pfad für die verkleinerte Miniaturansicht angelegt und dann diese Funktion aufgerufen wird, um ein Miniaturbild zu erstellen, nachdem das hochgeladene Bild verschoben wurde (Schritte 14 – 15).

1. `resize_image_gd()` hat vier Parameter:

- Pfad zum hochgeladenen Bild
- Pfad zum Speichern des skalierten Bilds
- Maximale Breite des neuen Bilds
- Maximale Höhe des neuen Bilds

2. Die GD-Funktion `getimagesize()` liefert ein Array mit Daten über das Bild zurück, einschließlich seiner Abmessungen und seines Medientyps (siehe vorherige Seite).

3. Breite, Höhe und Medientyp des Bilds werden aus dem Array gezogen und in Variablen gespeichert.

4. Die Variablen `$new_width` und `$new_height` werden mit der maximalen Breite und Höhe initialisiert, die das Miniaturbild haben darf.

5. `$orig_ratio` speichert das Seitenverhältnis des hochgeladenen Bilds.

6. Ist die Bildbreite größer als die Bildhöhe, so handelt es sich um ein Querformat.

7. Bei einem Bild im Querformat entspricht die Breite nach der Größenänderung der maximalen Bildbreite. Dieser Wert wurde in Schritt 4 beim Initialisieren der Variablen festgelegt.

Die neue Bildhöhe muss allerdings berechnet werden, indem die Bildbreite durch das Seitenverhältnis geteilt wird.

8. Andernfalls ist das Bild hochformatig oder quadratisch.

Seine Höhe entspricht der in Schritt 4 festgelegten maximalen Höhe.

Die neue Breite ergibt sich aus dem Produkt der neuen Bildhöhe mit dem Seitenverhältnis.

9. Mithilfe einer `switch`-Anweisung wird die richtige Funktion zum Öffnen des Bilds ausgewählt. (Wie auf der vorhergehenden Seite gezeigt, nutzt GD separate Funktionen zum Öffnen von Bildern mit unterschiedlichen Medientypen.)

Der Medientyp des Bilds (in Schritt 3 in `$media_type` gespeichert) wird als Bedingung für die `switch`-Anweisung verwendet. Das geöffnete Bild wird in `$orig` gespeichert.

10. Die GD-Funktion `imagecreatetruecolor()` erzeugt ein leeres Bild, das in `$new` gespeichert wird. Als Argumente werden die gewünschte Breite und Höhe des neuen Bilds übergeben.

11. Die GD-Funktion `imagecopyresampled()` kopiert das Originalbild, ändert seine Größe und fügt es in das in Schritt 10 erstellte neue Bild ein. Sie benötigt Werte für alle zehn auf der vorhergehenden Seite beschriebenen Parameter.

12. Eine weitere `switch`-Anweisung dient zur Auswahl der richtigen Speicherfunktion für das skalierte Bild. Diesmal verwenden wir eine Kurzform der `switch`-Anweisung (damit das Beispiel auf diese Seite passt). Die Funktionen zum Speichern von Bildern liefern `true` zurück, wenn das Bild gespeichert wurde, und `false`, wenn nicht; dieser Rückgabewert wird in `$result` gespeichert.

13. Der in `$result` gespeicherte Wert wird zurückgegeben.

14. Nachdem das Bild hochgeladen und verschoben wurde, wird der Pfad zum Speichern des neuen Miniaturbilds erstellt. Er setzt sich aus dem Pfad zum Ordner `uploads`, dem Text `thumb_` und dem Dateinamen zusammen.

15. `resize_image_gd()` wird aufgerufen.

```
<?php  
① function resize_image_gd($orig_path, $new_path, $max_width, $max_height)  
{  
②     $image_data = getimagesize($orig_path);           // Bilddaten abrufen  
③     $orig_width = $image_data[0];                     // Bildbreite  
④     $orig_height = $image_data[1];                    // Bildhöhe  
⑤     $media_type = $image_data['mime'];                // Medientyp  
⑥     $new_width = $max_width;                          // maximale neue Breite  
⑦     $new_height = $max_height;                        // maximale neue Höhe  
⑧     $orig_ratio = $orig_width / $orig_height;        // ursprüngliches Seitenverhältnis  
  
    // neue Größe berechnen  
    if ($orig_width > $orig_height) {                  // falls Querformat  
        $new_height = $new_width / $orig_ratio;          // neue Höhe gemäß Seitenverhältnis setzen  
    } else {  
        $new_width = $new_height * $orig_ratio;          // ansonsten  
    }  
  
    switch($media_type) {                                // Medientyp überprüfen  
        case 'image/gif' :  
            $orig = imagecreatefromgif($orig_path);       // falls GIF-Datei  
            break;  
        case 'image/jpeg' :  
            $orig = imagecreatefromjpeg($orig_path);      // öffnet diese Funktion das Bild  
            break;                                       // Switch-Statement verlassen  
        case 'image/png' :  
            $orig = imagecreatefrompng($orig_path);       // falls JPG-Datei  
            break;                                       // öffnet diese Funktion das Bild  
    }  
  
    $new = imagecreatetruecolor($new_width, $new_height); // leeres Bild erzeugen  
  
    ⑪ imagecopyresampled($new, $orig, 0, 0, 0, 0, $new_width, $new_height,  
                         $orig_width, $orig_height);           // Original in neues Bild kopieren  
  
    // Bild speichern - Miniaturbildordner muss schon bestehen + passende Zugriffsrechte haben  
    switch($media_type) {  
        case 'image/gif' : $result = imagegif($new, $new_path); break;  
        case 'image/jpeg': $result = imagejpeg($new, $new_path); break;  
        case 'image/png' : $result = imagepng($new, $new_path); break;  
    }  
    return $result;  
} ... // Code zum Hochladen und Validieren des Bilds wie auf den Seiten 296 bis 297  
$moved    = move_uploaded_file($_FILES['image']['tmp_name'], $destination); // Datei verschieben  
⑭ $thumbpath = $upload_path . 'thumb_' . $filename;                      // Miniaturbildordner anlegen  
⑮ $resized   = resize_image_gd($destination, $thumbpath, 200, 200); // Miniaturbild erstellen
```

SKALIEREN UND ZUSCHNEIDEN MIT IMAGICK

Mit der PHP-Extension Imagick können Sie die Open-Source-Bildbearbeitungssoftware ImageMagick über PHP-Code steuern. Die Imagick-Extension:

- benötigt viel weniger Code als GD
- berechnet Seitenverhältnisse und Abmessungen zur Größenanpassung von Bildern (Sie müssen diese nicht selbst in Ihrem Code berechnen)
- verwendet für alle Bildformate die gleichen Methoden
- unterstützt mehr Bildformate als GD

Allerdings müssen sowohl die Imagick-Extension als auch die ImageMagick-Software auf dem Webserver installiert sein, was standardmäßig nicht der Fall ist. Sie müssen also:

- für MAMP auf einem Mac Imagick aktivieren
- für XAMPP auf dem PC Imagick + ImageMagick installieren
- siehe <http://notes.re/php/install-imagick>
- prüfen, ob Ihr Webhosting-Anbieter dies unterstützt

Auf einem PC muss für Imagick zum Speichern einer Datei ein absoluter (kein relativer) Pfad angegeben werden, und absolute Pfade auf einem Windows-PC unterscheiden sich von denen auf einem Mac und unter Unix.

- auf Windows-PCs beginnen sie mit einem Laufwerkbuchstaben, z.B. C:/
- auf Mac- und Unix-Systemen beginnen sie mit einem Backslash \

Auch das Verzeichnis-Trennzeichen ist ein anderes: Auf dem PC ist es ein Schrägstrich, auf dem Mac und unter Unix ein Backslash. Um den korrekten Pfad zum Upload-Verzeichnis anzulegen (und ihn in einer Variablen zu speichern), nutzen Sie den folgenden Code.

Zur Verwendung von Imagick erstellen Sie mithilfe der Imagick-Klasse ein Objekt, das für das Bild steht, und übergeben dem Konstruktor dabei den Pfad zum gewünschten Bild.

VARIABLE ZUM SPEICHERN DES OBJEKTS	KLASSEN-NAME	PFAD ZUR BILDDATEI
\$image	= new	Imagick(\$filepath);

Das so erzeugte Imagick-Objekt verfügt über eine Reihe von Methoden zur Bearbeitung und Speicherung des Bilds.

METHODE	BESCHREIBUNG
thumbnailImage()	Bildgröße ändern
cropThumbnailImage()	Bild zuschneiden und skalieren
writeImage()	Bild speichern

Das nachfolgende Statement verwendet:

- Die PHP-Funktion dirname(), um den Pfad des Verzeichnisses zurückzuliefern, in dem sich die als Argument übergebene Datei befindet.
- Die Konstante __FILE__, die den Pfad zur derzeit ausgeführten Datei enthält.
- Die Konstante DIRECTORY_SEPARATOR, die das korrekte Verzeichnis-Trennzeichen für das die PHP-Datei ausführende Betriebssystem enthält.

AKTUELLE DATEI	ORDNER-TRENNZEICHEN
\$upload_path = dirname(__FILE__) . DIRECTORY_SEPARATOR . 'uploads'	. DIRECTORY_SEPARATOR;

PFAD ZUM ÜBERGEORDNETEN VERZEICHNIS

Die beiden nachfolgenden Beispiele umfassen zwei benutzerdefinierte Funktionen zur Größenänderung und zum Zuschneiden von Bildern mit Imagick.

Die Anweisungen, die diese Funktionen aufrufen, erscheinen direkt nach dem Code, der die hochgeladene Datei an ihren Zielort verschiebt, genau wie im letzten Beispiel auf Seite 305.

Der Code zum Hochladen, Überprüfen und Verschieben des Bilds ist derselbe wie auf den Seiten 296 und 297.

1. Die Funktion `create_thumbnail()` erstellt mittels Imagick eine Miniaturansicht eines Bilds. Die beiden Parameter lauten:

- Pfad zu dem soeben hochgeladenen Bild
 - Pfad für das neue von Imagick erstellte Miniaturbild
- 2.** Ein neues Objekt wird mit der Imagick-Klasse erstellt. Dazu ist der Pfad zum hochgeladenen Bild erforderlich.

3. Die Methode `thumbnailImage()` des Imagick-Objekts skaliert das Bild. Dazu verwendet sie drei Argumente:

- die neue Bildbreite
- die neue Bildhöhe
- den booleschen Wert `true`, um Imagick mitzuteilen, dass Breite und Höhe Maximalwerte sind und dass die Miniaturansicht das gleiche Seitenverhältnis haben soll wie das Original .

4. Die Imagick-Methode `writeImage()` speichert das Bild an dem im Parameter `$destination` spezifizierten Ort.

5. Die Funktion liefert `true` zurück, um anzugeben, dass der Vorgang erfolgreich war.

6. Nachdem die Datei verschoben wurde, wird in der Variablen `$thumbpath` der Pfad abgelegt, unter dem das neue Miniaturbild gespeichert wird.

7. `create_thumbnail()` wird mit dem Pfad des hochgeladenen Bilds und dem Pfad für die Miniaturansicht aufgerufen.

PHP

① function create_thumbnail(\$temporary, \$destination)

{

② \$image = new Imagick(\$temporary);
③ \$image->thumbnailImage(200, 200, true);
④ \$image->writeImage(\$destination);
⑤ return true;
} ... // Nachdem Datei validiert und verschoben wurde, Pfad zum Miniaturbild und dann Miniaturbild erstellen
⑥ \$moved = move_uploaded_file(\$_FILES['image']['tmp_name'], \$destination); // verschieben
⑦ \$thumbpath = \$upload_path . 'thumb_' . \$filename; // Pfad zum Miniaturbild
\$thumb = create_thumbnail(\$destination, \$thumbpath); // Miniaturbild erzeugen

// Objekt, das das Bild repräsentiert
// Miniaturbild erzeugen
// Datei speichern
// als Erfolgsmeldung true zurückliefern

section_b/c07/resize-im.php

8. `create_cropped_thumbnail()` erzeugt einen quadratischen Ausschnitt des hochgeladenen Bilds. Auf diese Weise wird sichergestellt, dass alle Miniaturansichten die gleiche Größe haben.

9. Der einzige Unterschied zum obigen Beispiel besteht darin, dass hier über die Methode `cropThumbnailImage()` des Imagick-Objekts eine beschnitten Miniaturansicht erstellt wird.

PHP

⑧ function create_cropped_thumbnail(\$temporary, \$destination)

{

⑨ \$image = new Imagick(\$temporary);
 \$image->cropThumbnailImage(200, 200, true);
 \$image->writeImage(\$destination);
 return true;
}

// Objekt, das das Bild repräsentiert
// Miniaturbild erzeugen
// Datei speichern
// als Erfolgsmeldung true zurückliefern

section_b/c07/crop-im.php

ZUSAMMENFASSUNG

BILDER & DATEIEN

- **HTML-Formulare nutzen ein Datei-Upload-Steuerelement zum Hochladen von Dateien.**
- **Das superglobale Array `$_FILES` speichert Daten über eine hochgeladene Datei.**
- **Hochgeladene Dateien werden an einem temporären Speicherort abgelegt. Sie müssen dann in einen anderen Ordner verschoben werden, um sie zu speichern.**
- **Bevor Sie versuchen, mit den Dateien zu arbeiten, vergewissern Sie sich, dass sie über HTTP hochgeladen wurden und dass keine Fehler auftreten sind.**
- **Stellen Sie sicher, dass der Dateiname nur zulässige Zeichen enthält.**
- **Überprüfen Sie vor dem Speichern die Größe und den Medientyp der hochgeladenen Dateien.**
- **Behalten Sie beim Skalieren eines Bilds das Seitenverhältnis bei, sonst wirkt es gestreckt und verzerrt.**
- **GD und Imagick sind Extensions, mit denen Sie die Bildgröße und den Bildausschnitt auf dem Server mittels PHP verändern können.**

8

DATUM &
UHRZEIT

Es gibt viele verschiedene Schreibweisen für Datums- und Uhrzeitangaben. In PHP integrierte Funktionen und Klassen erleichtern die Anzeige und Verarbeitung verschiedener Datums- und Uhrzeitformate.

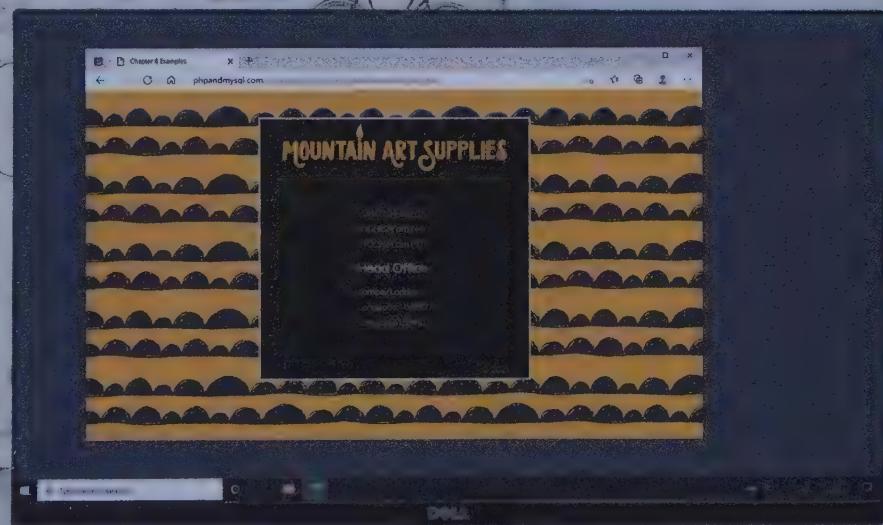
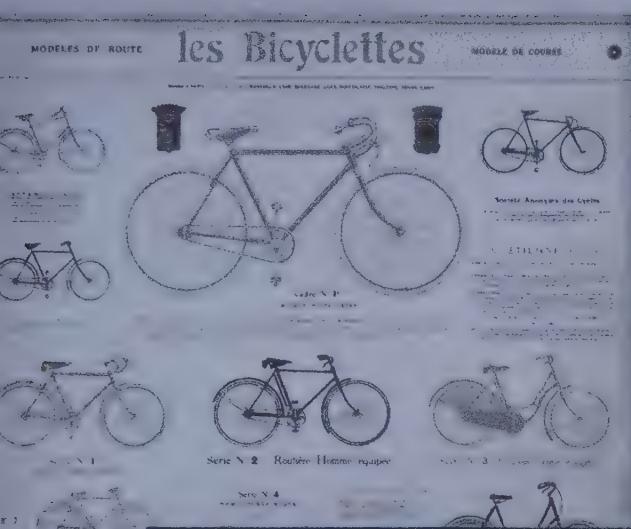
In diesem Kapitel lernen Sie die verschiedenen Varianten kennen, über die der PHP-Interpreter Datums- und Zeitangaben als Eingabe entgegennehmen und wie er diese für die Ausgabe auf einer Webseite formatieren kann. PHP kann folgende Datums- und Uhrzeitangaben verarbeiten:

- Datenelemente wie Jahre, Monate, Tage, Stunden, Minuten und Sekunden
- Formate wie 1. Juni 2001, 1/6/2001 oder nächster Dienstag
- Unix-Zeitstempel, die die Anzahl der Sekunden seit dem 1. Januar 1970 zählen (das erscheint vielleicht merkwürdig, aber viele Programmiersprachen verwenden diese Form der Zeitan-
gabe)

Nachdem Sie erst einmal erfahren haben, wie PHP mit Datums- und Zeitformaten umgeht, lernen Sie eine Reihe von eingebauten Funktionen kennen, die Unix-Zeitstempel generieren und wieder in ein für Menschen lesbaren Format zurückverwandeln.

Anschließend lernen Sie, wie Sie Datums- und Zeitangaben mithilfe von Objekten darstellen, die sich mit vier eingebauten Klassen erstellen lassen:

- `DateTime` erstellt Objekte, die ein bestimmtes Datum und eine bestimmte Uhrzeit darstellen
- `DateInterval` erstellt Objekte, die ein Zeitintervall darstellen (z.B. eine Stunde oder eine Woche)
- `DatePeriod` erstellt Objekte, die wiederkehrende Ereignisse darstellen, die in regelmäßigen Abständen stattfinden (z.B. jeden Tag, jeden Monat oder jedes Jahr)
- `DateTimeZone` erstellt Objekte, die eine Zeitzone darstellen



DATUMSFOMRAME

Datumsangaben lassen sich auf vielerlei Arten darstellen. PHP nutzt verschiedene Zeichen zur Beschreibung des Datumsformats.

Daten können aus folgenden Komponenten bestehen:

- Wochentag
- Tag des Monats
- Monat
- Jahr

PHP verwendet bestimmte Zeichen, um jede dieser Komponenten darzustellen. So steht etwa m-d-Y für das Datumsformat 04-06-2022. Diese Formatzeichen teilen dem PHP-Interpreter mit, wie Datumsangaben:

- beim Empfang verarbeitet werden
- für die Anzeige formatiert werden

Sie können zwischen den Formatzeichen Leerzeichen, Schrägstriche, Bindestriche und Punkte einfügen, um die einzelnen Komponenten optisch voneinander zu trennen.

Unten sehen Sie, wie die Formatzeichen unterschiedliche Schreibweisen für ein und dasselbe Datum definieren:

FORMATZEICHENFOLGE	DATUMSFOMRAME
I m j Y	Saturday April 6 2022
D j S F Y	Sat 6th April 2022
n/j/Y	4/6/2022
~y	04/06/22
m-d-Y	04-06-2022

WOCHENTAG

ZEICHEN	BESCHREIBUNG	BEISPIEL
D	die ersten drei Buchstaben	Sat
T	voller Name	Saturday

TAG DES MONATS

ZEICHEN	BESCHREIBUNG	BEISPIEL
0	Ziffern mit führender Null	09
1	Ziffern ohne führende Null	9
S	Endung	th

MONAT

ZEICHEN	BESCHREIBUNG	BEISPIEL
0	Ziffern mit führender Null	04
1	Ziffern ohne führende Null	4
M	die ersten drei Buchstaben	Apr
F	voller Name	April

JAHR

ZEICHEN	BESCHREIBUNG	BEISPIEL
Y	vier Ziffern	2022
y	zwei Ziffern	22

ZEITFORMATE

Mit diesen Formatzeichen können Sie unterschiedliche Darstellungsweisen für Zeitangaben definieren.

STUNDE

ZEICHEN	BESCHREIBUNG	BEISPIEL
h	12-Stunden-Anzeige mit führender Null	08
g	12-Stunden-Anzeige ohne führende Null	8
H	24-Stunden-Anzeige mit führender Null	08
G	24-Stunden-Anzeige ohne führende Null	8

MINUTE

ZEICHEN	BESCHREIBUNG	BEISPIEL
i	Ziffern mit führender Null	09

SEKUNDE

ZEICHEN	BESCHREIBUNG	BEISPIEL
s	Ziffern mit führender Null	04

AM/PM

ZEICHEN	BESCHREIBUNG	BEISPIEL
a	Kleinbuchstaben	am
A	Großbuchstaben	AM

Zeitangaben können aus folgenden Komponenten bestehen:

- Stunden
- Minuten
- Sekunden
- am/pm (falls kein 24-Stunden-Format genutzt wird)

Diese einzelnen Elemente lassen sich alle durch Formatzeichen darstellen. Zum Beispiel steht g:i a für eine Uhrzeit im Format 8:09 am. Mit diesen Formatzeichen wird dem PHP-Interpreter mitgeteilt, wie die Zeiten:

- beim Empfang verarbeitet werden
- für die Anzeige formatiert werden

Sie können zwischen den Formatzeichen Leerzeichen, Doppelpunkte, Punkte und Klammern einfügen, um die einzelnen Komponenten optisch voneinander zu trennen.

Unten sehen Sie, wie die Formatzeichen unterschiedliche Schreibweisen für ein und dieselbe Uhrzeit definieren:

FORMATZEICHEN-FOLGE	ZEITFORMAT
g:i a	8:09 am
08:09(A)M	08:09
6:1	08:09

DATUMS- & UHRZEIT- ANGEBEN MITTELS STRING

Einige Funktionen und Methoden erlauben die Angabe von Datum und Uhrzeit in Form einer Zeichenkette. Der String muss dazu in einem zulässigen Format vorliegen (siehe unten).

Der PHP-Interpreter kann Datumsangaben in den folgenden String-Formaten verarbeiten. Wenn Schrägstriche verwendet werden, erwartet der PHP-Interpreter den Monat vor dem Tag des Monats. Kommen Bindestriche oder Punkte zum Einsatz, erwartet er den Tag des Monats vor dem Monat.

DATUMSFOR- MAT	BEISPIEL
d F Y	04 September 2022
JS F Y	4th September 2022
F j Y	September 4 2022
l d Y	Sep 04 2022
m/d/Y	09/04/2022
Y/m/d	2022/09/04
d-m-Y	04-09-2022
n-j-Y	9-4-2022
d.m.y	04.09.22

Sie können die auf der rechten Seite vorgestellten relativen Zeitangaben verwenden, z.B.:

+1 day

-1 year +2 days, 1 month

+4 hours 20 mins

next Tuesday

first Sat of Jan

First/Last funktioniert nur für Tage des Monats. Wurde keine Uhrzeit angegeben, wird sie auf Mitternacht gesetzt.

Der PHP-Interpreter akzeptiert Zeitangaben in den unter stehenden String-Formaten. Sie können auch:

- Groß- oder Kleinbuchstaben für am und pm verwenden
- mit dem Buchstaben t ein Datum von einer Uhrzeit trennen
- eine Zeitzone dahinter einfügen

12-STUNDEN-ZEITFORMAT	BEISPIEL
-----------------------	----------

g:i	4am
g:i:a	4:08 am
g:i:s a	4:08:37 am
g:i:s A	4.08.37 am

24-STUNDEN-ZEITFORMAT	BEISPIEL
-----------------------	----------

H:i	04:08
H:i:s	04:08:37
H:i:s	040837
H.i.s	04.08.37

TYP	RELATIVE ZEITANGABE
-----	---------------------

Addieren/subtrahieren	+
Betrag	0 - 9
Zeiteinheiten (auch im Plural)	day, fortnight, month, year, hour, min, minute, sec, second
Tagesnamen	Monday - Sunday and Non - Sun
Relative Angaben	next, last, previous, this
Ordnungszahlen	first - twelfth

UNIX-ZEITSTEMPEL

Unix-Zeitstempel geben Daten und Uhrzeiten durch die Anzahl der Sekunden an, die seit dem 1. Januar 1970 um Mitternacht verstrichen sind.

DATUM	UHRZEIT	UNIX-ZEITSTEMPEL
31 DEC 1969	+  = - 60 23:59:00	- 60
1 JAN 1970	+  = 120 00:02:00	120
11 APR 1975	+  = 166878000 11:00:00	166878000
30 AUG 2000	+  = 967644000 14:00:00	967644000
31 DEC 2020	+  = 1609426800 15:00:00	1609426800

Mit dem PHP-Interpreter können Sie Datums- und Zeitangaben in Form von Unix-Zeitstempeln angeben und abrufen.

Links sehen Sie einige konkrete Beispiele für Datums- und Zeitangaben, gefolgt von den entsprechenden Unix-Zeitstempeln.

Daten vor dem 1. Januar 1970 werden mit negativen Zahlen ausgedrückt.

Wie Sie sehen werden, helfen Ihnen die in PHP eingebauten Funktionen und Klassen beim Umgang mit Unix-Zeitstempeln. Sie verwenden die Formatzeichen, die Sie gerade kennengelernt haben, um zu definieren, wie die Funktionen und Klassen einen Unix-Zeitstempel in ein für Menschen lesbares Format umwandeln sollen.

Das späteste mit einem Unix-Zeitstempel darstellbare Datum ist der 19. Januar 2038.

Das Betriebssystem Unix wurde in den 1970er-Jahren entwickelt.

INTEGRIERTE DATUMS- UND UHRZEITFUNKTIONEN

PHP verfügt über eingebaute Funktionen zum Erzeugen von Unix-Zeitstempeln zur Darstellung von Datum und Uhrzeit. Außerdem verfügt es über integrierte Funktionen zur Konvertierung dieser Unix-Zeitstempel in ein leicht lesbares Format.

Die drei nachstehenden Funktionen dienen alle der Erstellung eines Unix-Zeitstempels.

FUNKTION

`time()`

`strtotime($string)`

`mktime(H, i, s, n, j, Y)`

`date()` wandelt Unix-Zeitstempel in ein für Menschen lesbares Format um.

FUNKTION

`date($format[, $timestamp])`

Wenn die Erstellung eines Zeitstempels nicht möglich ist, geben sie `false` zurück.

BESCHREIBUNG

Liefert das aktuelle Datum und die Uhrzeit als Unix-Zeitstempel zurück.

Wandelt einen String in einen Unix-Zeitstempel um (akzeptiert die auf Seite 314 angegebenen Formate).

BEISPIEL

```
strtotime('December 1 2020');
```

```
strtotime('1/12/2020');
```

Wandelt Datums-/Zeitkomponenten (in den Argumenten) in einen Unix-Zeitstempel um.

BEISPIEL

```
mktime(17, 01, 05, 2, 1,  
2001);
```

```
mktime(01, 30, 45, 4, 29,  
2020);
```

STEHT FÜR

February 12 2001 17:01:05

April 29 2020 01:30:45

Das Format wird mit den Formatzeichen auf den Seiten 312 und 313 vorgegeben.

BESCHREIBUNG

Wenn kein Zeitstempel angegeben wird, werden das aktuelle Datum und die aktuelle Uhrzeit angezeigt.

Wandelt einen Unix-Zeitstempel in eine für Menschen lesbare Form um: Der erste Parameter gibt an, wie das Datum formatiert werden soll. Der zweite Parameter ist der zu formatierende Unix-Zeitstempel.

BEISPIEL

```
date('Y');
```

```
date('d-m-y h:i a',  
1609459199);
```

```
date('D j M Y H:i:a',  
1609459199);
```

AUSGABE

aktuelles Jahr

31-12-20 11:59 pm

Thu 31 Dec 2020
23:59:59

DATUMSFUNKTIONEN

PHP

section_b/c08/date-functions.php

```
<?php  
① $start      = strtotime('January 1 2021');  
② $end        = mktime(0, 0, 0, 2, 1, 2021);  
③ [ $start_date = date('l, d M Y', $start);  
③ [ $end_date   = date('l, d M Y', $end);  
?>  
<?php include 'includes/header.php'; ?>  
  
④ [ <p><b>Sale starts:</b> <?= $start_date ?></p>  
④ [ <p><b>Sale ends:</b> <?= $end_date ?></p>  
  
<?php include 'includes/footer.php'; ?>
```

PHP

section_b/c08/includes/footer.php

⑤ <footer>© <?php echo date('Y')?></footer> ...

ERGEBNIS



1. Die Funktion `strtotime()` erzeugt einen Unix-Zeitstempel, der ein Datum in der Vergangenheit darstellt. Dieser wird in der Variablen `$start` gespeichert.

2. Die Funktion `mktime()` erzeugt einen Unix-Zeitstempel, der ein Datum einen Monat später darstellt.

Er wird in `$end` gespeichert.

3. Die Funktion `date()` wandelt diese Unix-Zeitstempel in ein lesbares Format um, bestehend aus dem:

- Tagesnamen
- Tag (mit führender Null)
- Monat (erste drei Buchstaben)
- Jahr (vier Ziffern)

Die Datumsangaben werden in den Variablen `$start_date` und `$end_date` gespeichert.

4. Die für Menschen lesbare Version der beiden Daten wird angezeigt.

5. Die Include-Datei für die Fußzeile fügt einen Copyright-Hinweis ein. Über die Funktion `date()` wird das Jahr ausgegeben. Da kein Zeitstempel übergeben wird, wird das aktuelle Jahr verwendet.

Probieren Sie es: Ändern Sie in Schritt 2 das Datum und die Uhrzeit auf nächste Woche um 12 Uhr. Ändern Sie in Schritt 3 das Datumsformat auf Mon 1st February 2021.

Hinweis: Wenn die Zeit um ein paar Stunden abweicht, überprüfen Sie die Standardeinstellung der Zeitzone in der Datei `php.ini` (siehe Seite 198).

OBJEKTE ZUR REPRÄSENTATION VON DATUMS- UND ZEITANGABEN

Die in PHP integrierte DateTime-Klasse erzeugt ein Objekt, das ein Datum und eine Uhrzeit abbildet. Dessen Methoden können das Datum und die Uhrzeit, die das Objekt darstellt, in einem für Menschen lesbaren Format oder als Unix-Zeitstempel zurückliefern.

Um ein DateTime-Objekt zu erstellen, verwenden Sie:

- eine Variable zur Unterbringung des Objekts
- den Zuweisungsoperator
- das Schlüsselwort new
- den Klassennamen DateTime
- ein Klammernpaar

In den Klammern geben Sie das Datum und die Uhrzeit an, die das Objekt repräsentieren soll.

Sie können jedes der auf Seite 314 beschriebenen Datums- und Uhrzeitformate verwenden. Der Wert sollte in Anführungszeichen gesetzt werden.

Wenn Sie kein Datum und keine Uhrzeit angeben, greift das Objekt auf das aktuelle Datum und die aktuelle Uhrzeit zurück.

Wenn Sie ein Datum, aber keine Uhrzeit angeben, verwendet das Objekt Mitternacht des angegebenen Tages.

```
$date = new DateTime('2001-02-01 15:01:05');
```

VARIABLE

KLASSENNAME

DATUM UND UHRZEIT

Sie können auch die Funktion date_create_from_format() verwenden, um ein DateTime-Objekt zu erzeugen.

Das erste Argument ist das Format, in dem das Datum und die Uhrzeit angegeben werden.

Das zweite Argument ist das Datum und die Uhrzeit in dem angegebenen Format. Beide Argumente müssen in Anführungszeichen stehen.

```
$date = date_create_from_format('j-M-Y', '15-Jan-2020');
```

VARIABLE

FUNKTION

FORMAT

DATUM/UHRZEIT

Die folgenden Methoden des DateTime-Objekts geben das Datum und die Uhrzeit zurück, die das Objekt darstellt.

Um Datum und Uhrzeit in einem für Menschen lesbaren Format zu erhalten, verwenden Sie die Methode format().

Um Datum und Uhrzeit als Unix-Zeitstempel zu erhalten, verwenden Sie die Methode getTimestamp().

METHODE

BESCHREIBUNG

```
format($format[, $Date  
Timezone])
```

Ruft das Datum und die Uhrzeit im angegebenen Format ab.
Der optionale zweite Parameter legt eine Zeitzone fest (siehe Seite 326).

```
getTimestamp()
```

Gibt den Unix-Zeitstempel für das Datum und die Uhrzeit zurück, die das Objekt darstellt.

DATETIME - OBJEKT

PHP

section_b/c08/datetime-object.php

```
<?php  
① $start = new DateTime('2021-01-01 00:00');  
② $end   = date_create_from_format('Y-m-d H:i',  
    '2021-02-01 00:00');  
?  
<?php include 'includes/header.php'; ?>  
  
<p><b>Sale starts:</b>  
③ <?= $start->format('l, jS M Y H:i') ?></p>  
<p><b>Sale ends:</b>  
④ <?= $end->format('l, jS M Y') ?> <b>at</b>  
⑤ <?= $end->format('H:i') ?></p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. In diesem Beispiel wird zu- nächst mit der `DateTime`-Klasse ein Objekt erzeugt und in der Variablen `$start` gespeichert.

2. Ein zweites `DateTime`-Objekt wird mit der Funktion `date_create_from_format()` erstellt. Der erste Parameter gibt das gewünschte Datumsformat an. Der zweite Parameter legt das Datum und die Uhrzeit fest. Dieses Objekt wird in der Variablen `$end` gespeichert.

3. Das Startdatum und die Uhrzeit werden mit der Methode `format()` des `DateTime`-Objekts auf der Seite ausgegeben. Das Argument gibt an, in welchem Format das Datum und die Uhrzeit ausgegeben werden sollen.

4. Das Enddatum (nicht die Uhrzeit) wird mit der Methode `format()` des `DateTime`-Objekts auf der Seite ausgegeben. Ihr Parameter gibt das Ausgabeformat für das Datum an.

5. Die Endzeit wird separat ausgegeben und bedient sich dabei ebenfalls der `format()`-Methode. Wie Sie sehen, können Sie das Datum oder die Uhrzeit im Objekt also auch einzeln ausgeben.

Probieren Sie es: Setzen Sie das Datum in Schritt 1 auf den gestrigen Tag. In Schritt 2 ändern Sie das Datum auf 7 Tage nach Angebotsbeginn.

DATUM & UHRZEIT IN DATETIME-OBJEKten AKTUALISIEREN

Sobald Sie ein Objekt mit der DateTime-Klasse erstellt haben, können Sie mithilfe der nachstehenden Methoden das Datum/die Uhrzeit, das/die es darstellt, festlegen oder aktualisieren.

Die Methoden zum Setzen von Datum und Uhrzeit überschreiben die möglicherweise aktuell vom Objekt dargestellten Zeit- und Datumsangaben.

Die Methoden add() und sub() nutzen ein DateInterval-Objekt, das auf Seite 322 vorgestellt wird.

METHODE	BESCHREIBUNG
setDate(\$year, \$month, \$day)	Setzt ein Datum für das Objekt
setTime(\$hour, \$minute [, \$seconds][, \$microseconds])	Setzt eine Uhrzeit für das Objekt
setTimestamp(\$timestamp)	Setzt das Datum/die Uhrzeit mit einem Unix-Zeitstempel
modify(\$DateFormat)	Aktualisiert das Datum/die Uhrzeit mit einer Zeichenkette
add(\$DateInterval)	Addiert mithilfe eines DateInterval-Objekts ein Zeitintervall (siehe Seite 322)
sub(\$DateInterval)	Subtrahiert mithilfe eines DateInterval-Objekts ein Zeitintervall (siehe Seite 322)

Wenn der PHP-Interpreter eine Variable erstellt, kann er darin einen Einzelwert oder ein Array speichern.

Wenn der PHP-Interpreter ein Objekt erstellt, legt er dieses an einem unabhängigen Ort in seinem Speicher ab. Wird dieses Objekt dann in einer Variablen gespeichert, enthält die Variable einen Verweis auf diesen Speicherort (anstelle des Objekts selbst).

Das heißt, wenn Sie ein Objekt erstellen und es in einer Variablen speichern und Sie dann noch eine zweite Variable deklarieren und dieser das gleiche Objekt als Wert zuweisen, würden beide Variablen den Speicherort des gleichen Objekts enthalten.

Wenn Sie also das Objekt in einer Variablen aktualisieren, wird es auch in der anderen aktualisiert:

```
$start = new DateTime('2020/12/1');
$end   = $start;
// beide Variablen verweisen auf dasselbe Objekt
$end->modify('+1 day');

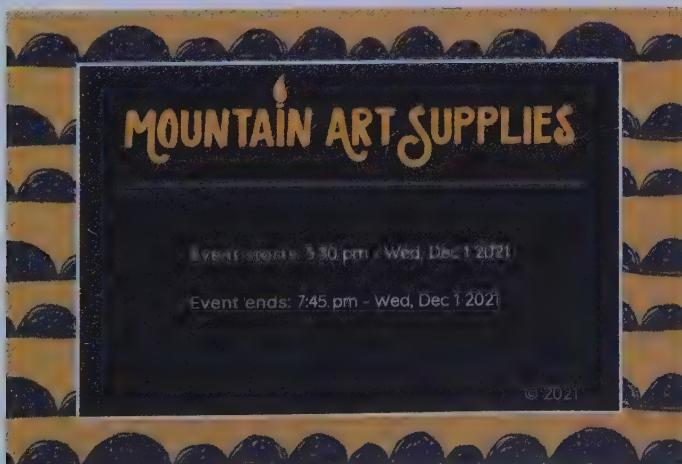
Um dies zu umgehen, können Sie mit dem Schlüsselwort clone eine Kopie des Objekts erstellen:
$start = new DateTime('2020/12/1');
$end   = clone $start;
// nur das Objekt in $end wird verändert
$end->modify('+1 day');
```

SO SETZEN SIE DATUM & UHRZEIT IN EINEM DATETIME-OBJEKT

PHP section_b/c08/datetime-object-set-date-and-time.php

```
<?php  
① $start = new DateTime();  
② $start->setDate(2021, 12, 01);  
③ $start->setTime(17, 30);  
④ $end = clone $start;  
⑤ $end->modify('+2 hours 15 min');  
?>  
<?php include 'includes/header.php'; ?>  
  
<p><b>Event starts:</b>  
⑥ <?= $start->format('g:i a - D, M j Y') ?></p>  
  
<p><b>Event ends:</b>  
⑥ <?= $end->format('g:i a - D, M j Y') ?></p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Mit der Klasse `DateTime` wird ein neues Objekt erstellt und in der Variablen `$start` gespeichert. Es enthält das aktuelle Datum und die aktuelle Uhrzeit.

2. Das Datum wird mit der Methode `setDate()` des `DateTime`-Objekts gesetzt.

3. Die Uhrzeit wird mit der Methode `setTime()` des `DateTime`-Objekts aktualisiert.

4. Das in `$start` gespeicherte Objekt wird mit dem Schlüsselwort `clone` geklont und der Klon in der Variablen `$end` gespeichert.

5. Mit der `modify()`-Methode des `DateTime`-Objekts wird das in `$end` gespeicherte Objekt so aktualisiert, dass es ein Datum und eine Uhrzeit darstellt, die 2 Stunden und 15 Minuten nach dem Datum und der Uhrzeit im in `$start` gespeicherten Objekt liegen.

6. Die von den beiden Objekten repräsentierten Datums- und Zeitangaben werden mit der Methode `format()` ausgegeben.

Probieren Sie es: Ändern Sie in Schritt 5 das Ende des Events so, dass es zwei Tage hinter dem Startdatum liegt.

EINEN ZEITRAUM MITTELS DATEINTERVAL DARSTELLEN

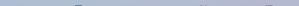
Die Klasse DateInterval wird zur Erzeugung eines Objekts verwendet, das ein in Jahren, Monaten, Wochen, Tagen, Stunden, Minuten und Sekunden bemessenes Zeitintervall darstellt.

Die Methoden `add()` und `sub()` des `DateTime`-Objekts nutzen ein `DateInterval`-Objekt zu Angabe einer Zeitspanne, die zum aktuellen Datum/Zeitpunkt hinzugefügt oder davon abgezogen werden soll. Die Angabe des Zeitraums erfolgt dabei im rechts dargestellten Format.

Jedem Intervall wird der Buchstabe P vorangestellt. Der Buchstabe T steht vor einer Zeitspanne mit Stunden-, Minuten- oder Sekundenangabe.

INTERVALL	SCHREIBWEISE
1 Jahr	P1Y
2 Monate	P2M
3 Tage	P3D
1 Jahr, 2 Monate, 3 Tage	P1Y2M3D
1 Stunde	PT1H
30 Minuten	PT30M
15 Sekunden	PT15S
1 Stunde, 30 Minuten, 15 Sekunden	PT1H30M15S
1 Jahr, 1 Tag, 1 Stunde und 30 Minuten	P1Y1DT1H30M

```
$interval = new DateInterval('P1M');
```



Die `diff()`-Methode des `DateTime`-Objekts (kurz für Differenz) vergleicht zwei `DateTime`-Objekte und gibt ein `DateInterval`-Objekt zurück, das die dazwischen liegende Zeitspanne darstellt.

Um das in einem DateInterval-Objekt gespeicherte Intervall anzuzeigen, nutzen Sie dessen format()-Methode. Ihr Argument ist ein String, der die rechts angegebenen Formatzeichen verwendet.

```
$interval->format('%h hours %i minutes');  
          ^    ^  
        INTERVAL  INTERVAL
```

INTERVALL	BESCHREIBUNG
%y	Jahre
%m	Monate
%d	Tage
%h	Stunden
%i	Minuten
%s	Sekunden
%f	Mikrosekunden

DATEINTERVAL-OBJEKT

PHP

section_b/c08/dateinterval-object.php

```
<?php  
① $today      = new DateTime();  
② $event       = new DateTime('2025-12-31 20:30');  
③ $countdown = $today->diff($event);  
  
④ $earlybird = new DateTime();  
⑤ $interval   = new DateInterval('P1M');  
⑥ $earlybird->add($interval);  
?  
<?php include 'includes/header.php'; ?>  
  
<p><b>Countdown to event:</b><br>  
⑦  <?= $countdown->format('%y years %m months %d days') ?>  
</p>  
<p><b>50% off tickets bought by:</b><br>  
⑧  <?= $earlybird->format('D d M Y, g:i a') ?>  
</p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Das aktuelle Datum und die aktuelle Uhrzeit werden durch ein DateTime-Objekt abgebildet und in \$today gespeichert.

2. Ein Veranstaltungsdatum wird durch ein DateTime-Objekt dargestellt und in der Variablen \$event gespeichert.

3. Die diff()-Methode des DateTime-Objekts ermittelt das Zeitintervall zwischen dem aktuellen Zeitpunkt und dem Veranstaltungsdatum.

Das zurückgelieferte DateInterval-Objekt wird in \$countdown gespeichert.

4. Das aktuelle Datum und die aktuelle Uhrzeit werden in \$earlybird gespeichert.

5. Ein Intervall von einem Monat wird durch ein DateInterval-Objekt dargestellt und in \$interval gespeichert.

6. Die add()-Methode des DateTime-Objekts addiert das Intervall im DateInterval-Objekt zum aktuellen Datum in \$earlybird.

7. Das in \$countdown gespeicherte Zeitintervall wird ausgegeben. Achten Sie auf das %-Symbol vor den Formatzeichen, die die Intervalle darstellen.

8. Das in \$earlybird gespeicherte Datum wird ausgegeben.

Probieren Sie es: Verschieben Sie den Veranstaltungstermin in Schritt 2 um 3 Monate in die Zukunft. Ändern Sie das Intervall in Schritt 5 auf 12 Stunden.

WIEDERKEHRENDE EREIGNISSE MIT DATEPERIOD

Die DatePeriod-Klasse kann ein Objekt erstellen, das eine Reihe von DateTime-Objekten enthält, die in regelmäßigen Abständen zwischen einem Anfangs- und Enddatum auftreten. Sie können die so erstellten einzelnen DateTime-Objekte dann in einer Schleife durchlaufen.

Zur Erstellung eines DatePeriod-Objekts benötigen Sie drei Dinge:

- ein Anfangsdatum (DateTime-Objekt)
 - die Häufigkeit des Ereignisses (ein DateInterval-Objekt)
 - ein Enddatum für den Zeitraum

Das Enddatum für den Zeitraum kann entweder sein:

- ein DateTime-Objekt oder
 - eine ganze Zahl, die angibt, wie oft das Ereignis (nach dem Anfangsdatum) auftreten soll

Ein neu erstelltes DatePeriod-Objekt enthält eine Reihe von Date-Time-Objekten, von denen jedes einen Zeitpunkt zwischen dem Anfangs- und Enddatum mit dem im DateInterval-Objekt angegebenen Zeitabstand darstellt.

`$period = new DatePeriod($start, $interval, $end);`

<code>\$period</code>	<code>DatePeriod</code>	<code>\$start</code>	<code>\$interval</code>	<code>\$end</code>
VARIABLE	KLASSEN-NAME	DATUM/UHRZEIT (START)	INTERVALL	DATUM/UHRZEIT (ENDE)

Mit einer `foreach`-Schleife können Sie auf jedes in einem `DatePeriod`-Objekt enthaltene `DateTime`-Objekt zugreifen.

Wie bei allen Schleifen nutzen Sie einen Variablennamen, um die einzelnen DateTime-Objekte während der Schleifendurchläufe zu speichern.

Im Codeblock der Schleife können Sie die Methoden des `DateTime`-Objekts verwenden, um mit dessen Datum/Uhrzeit zu arbeiten.

```
DATEPERIOD-OBJEKT ENTHÄLT          VARIABLENNNAME ZUR REPRÄSENTATION DER DATETIME-OBJEKTE  
DATETIME-OBJEKTE  
  
foreach($period as $occurrence) {  
    echo $occurrence->format('Y jS F');  
}
```

DATEPERIOD-OBJEKT

PHP

section_b/c08/dateperiod-object.php

```
<?php  
① $start = new DateTime('2025-1-1');  
② $end = new DateTime('2026-1-1');  
③ $interval = new DateInterval('P1M');  
④ $period = new DatePeriod($start, $interval, $end);  
?>  
<?php include 'includes/header.php'; ?>  
  
<p>  
⑤ <?php foreach ($period as $event) { ?>  
    <b><?= $event->format('l') ?></b>,  
    <?= $event->format('M j Y') ?></b><br>  
    <?php } ?>  
</p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



1. Die Variable `$start` enthält ein DateTime-Objekt, das den 1. Januar 2025 darstellt.
2. Die Variable `$end` enthält ein DateTime-Objekt, das den 1. Januar 2026 darstellt.
3. Die Variable `$interval` enthält ein DateInterval-Objekt, das einen Monat darstellt.
4. Die Variable `$period` enthält das DatePeriod-Objekt. Dieses benötigt drei Parameter (die Werte wurden in den Schritten 1-3 definiert):
 - ein Anfangsdatum
 - ein Intervall
 - ein Enddatum

Es wird zwölf DateTime-Objekte enthalten (eines für jeden Monat des Jahrs 2025).

5. Eine `foreach`-Schleife durchläuft jedes der DateTime-Objekte. Innerhalb der Schleife steht `$event` für jedes DateTime-Objekt. Für jedes dieser Objekte:
 6. Die Methode `format()` gibt den Wochentag und dann den Monat, den Tag und das Jahr aus.

Probieren Sie es: Ändern Sie das Intervall in Schritt 3 auf drei Monate (`P3M`).

ZEITZONEN VERWALTEN MIT DATETIMEZONE

Bei der Erstellung eines DateTime-Objekts können Sie eine Zeitzone angeben. Diese wird durch ein Objekt dargestellt, das mit der Klasse DateTimeZone erzeugt wird.

Die Klasse DateTimeZone erstellt ein Objekt, das eine Zeitzone darstellt. Es speichert Informationen über diese Zeitzone.

In den Klammern geben Sie die Zeitzone als IANA-Zeitzone an (eine vollständige Liste finden Sie unter <http://notes.re/timezones>).

Sie können dieses Objekt bei der Erzeugung eines DateTime-Objekts verwenden, um dessen Zeitzone zu definieren, die auch Steuerung von Sommer- und Winterzeit übernimmt.

```
VARIABLE           KLASSENNAME          ZEITZONE
$tz_LDN = new DateTimeZone('Europe/London');
$LDN = new DateTime('now', $tz_LDN);
                                |
                                |
DATETIMEZONE-OBJEKT
```

METHODE	BESCHREIBUNG										
getName()	Liefert den Namen dieser Zeitzone zurück										
getLocation()	Liefert ein indiziertes Array mit den folgenden Informationen zurück: <table><thead><tr><th>SCHLÜSSEL</th><th>WERT</th></tr></thead><tbody><tr><td>country_code</td><td>Kurzcode für das Land</td></tr><tr><td>latitude</td><td>Breitengrad dieses Orts</td></tr><tr><td>longitude</td><td>Längengrad dieses Orts</td></tr><tr><td>comments</td><td>Kommentare zu diesem Ort</td></tr></tbody></table>	SCHLÜSSEL	WERT	country_code	Kurzcode für das Land	latitude	Breitengrad dieses Orts	longitude	Längengrad dieses Orts	comments	Kommentare zu diesem Ort
SCHLÜSSEL	WERT										
country_code	Kurzcode für das Land										
latitude	Breitengrad dieses Orts										
longitude	Längengrad dieses Orts										
comments	Kommentare zu diesem Ort										
getOffset()	Liefert die Abweichung von der UTC für diese Zeitzone in Sekunden zurück (die UTC-Zeit entspricht der GMT, aber sie ist standardisiert und nicht an ein Land/Territorium gebunden).										
getTransitions()	Gibt ein Array zurück, das angibt, wann in der angegebenen Zeitzone die Sommerzeit in Kraft tritt.										

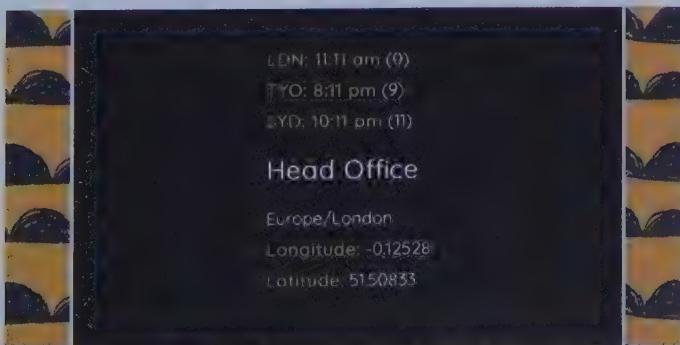
DATETIMEZONE-OBJEKT

PHP

section_b/c08/datetimezone-object.php

```
<?php  
① [ $tz_LDN = new DateTimeZone('Europe/London');  
② [ $tz_TYO = new DateTimeZone('Asia/Tokyo');  
③ [ $LDN = new DateTime('now', $tz_LDN);  
④ [ $TYO = new DateTime('now', $tz_TYO);  
⑤ [ $SYD = new DateTime('now',  
    new DateTimeZone('Australia/Sydney'));  
?> ...  
<p><b>LDN: <?= $LDN->format('g:i a') ?></b>  
(<?= ($LDN->getOffset() / (60 * 60)) ?>)<br>  
<b>TYO: <?= $TYO->format('g:i a') ?></b>  
(<?= ($TYO->getOffset() / (60 * 60)) ?>)<br>  
<b>SYD: <?= $SYD->format('g:i a') ?></b>  
(<?= ($SYD->getOffset() / (60 * 60)) ?>)<br></p>  
  
⑥ <h1>Head Office</h1>  
⑦ [ <p><?= $tz_LDN->getName() ?><br>  
[ <b>Longitude:</b> <?= $location['longitude'] ?><br>  
[ <b>Latitude:</b> <?= $location['latitude'] ?></p>
```

ERGEBNIS



1. Zwei DateTimeZone-Objekte werden erstellt, um die Zeitzonen für London und Tokio wiederzuspiegeln.

2. getLocation() liefert die Standortdaten für die Zeitzone London als Array. Das Array wird in \$location gespeichert.

3. Zwei DateTime-Objekte werden unter Verwendung der DateTimeZone-Objekte aus Schritt 1 erstellt. Sie repräsentieren die aktuelle Zeit in zwei Zeitzonen.

4. Ein drittes DateTime-Objekt wird erstellt, um zu zeigen, wie das DateTimeZone-Objekt zusammen mit dem DateTime-Objekt erstellt werden kann.

5. Für jedes der DateTime-Objekte:

- format() zeigt die aktuelle Zeit an diesem Ort an.
- getOffset() zeigt die Zeitdifferenz zwischen den Orten und UTC an. Die Funktion liefert den Zeitunterschied in Sekunden, der durch (60 * 60) geteilt wird, um die Abweichung in Stunden anzuzeigen.

6. Der Name der ersten Zeitzone wird mit getName() abgerufen.

7. Der Längen- und der Breitengrad der Zeitzone werden ausgegeben.

Probieren Sie es: Erstellen Sie ein Objekt für eine Niederlassung in Los Angeles und zeigen Sie die dortige Uhrzeit an.

ZUSAMMENFASSUNG

DATUM & UHRZEIT

- Mit Formatzeichen können Sie angeben, wie ein Datum oder eine Uhrzeit formatiert werden sollen.
- Unix-Zeitstempel stellen ein Datum und eine Uhrzeit anhand der seit dem 1. Januar 1970 verstrichenen Sekunden dar.
- Die Funktionen `time()`, `strptime()` und `mktime()` erzeugen Unix-Zeitstempel. Die Funktion `date()` wandelt einen Unix-Zeitstempel in ein für Menschen lesbares Format um.
- Die Klasse `DateTime` erstellt Objekte, die Datums- und Zeitangaben repräsentieren. Sie verfügt über Methoden zum Verändern von Datums- und Zeitangaben und zur Darstellung in von Menschen lesbaren Formaten.
- Die Klasse `DateInterval` erstellt Objekte, die ein Zeitintervall repräsentieren, z.B. einen Monat oder ein Jahr.
- Die Klasse `DatePeriod` erstellt eine Reihe von `DateTime`-Objekten, die wiederkehrende Ereignisse repräsentieren.
- Die Klasse `DateTimeZone` erstellt Objekte, die Zeitzonen repräsentieren und Informationen über diese enthalten.

9

COOKIES & SITZUNGEN

Zur Erstellung von Webseiten mit personenbezogenen Daten wie Benutzername, Profilbild oder einer Liste der zuletzt besuchten Seiten muss die Website wissen, wer die jeweilige Seite anfordert.

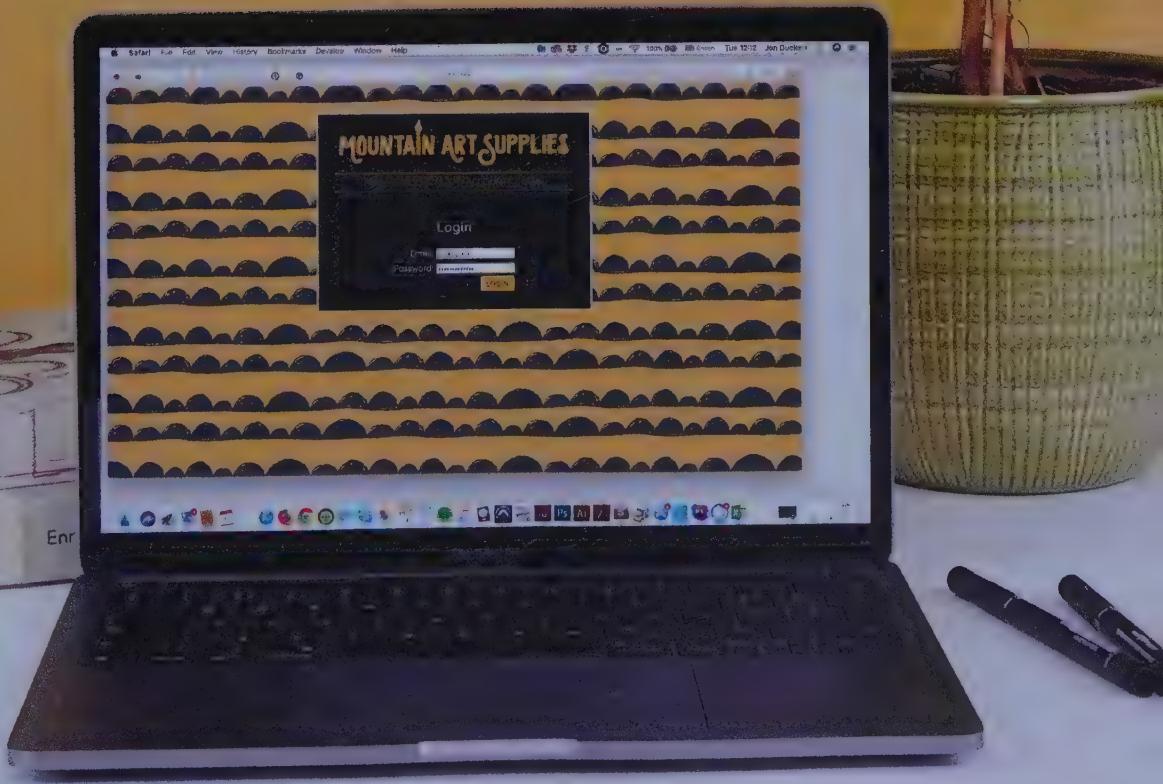
Die Regeln des HTTP-Protokolls legen fest, wie ein Browser eine Webseite anfordert und wie der Server darauf antwortet, aber es behandelt jede Anforderung und jede Antwort separat. HTTP stellt keinen Mechanismus bereit, mit dem eine Website feststellen kann, welcher Besucher eine Seite anfordert.

Wenn eine Website wissen muss, wer eine Webseite anfordert, oder wenn sie personalisierte Informationen anzeigen soll, kann sie die einzelnen Besucher erfassen und Informationen über ihre Vorlieben speichern. Dies erfolgt über eine Mischung aus Cookies und Sitzungen.

- Cookies sind Textdateien, die im Browser gespeichert werden. Eine Website kann dem Browser mitteilen, welche Daten in einem Cookie gespeichert werden sollen, und der Browser sendet diese Daten dann bei jeder nachfolgenden Seitenanforderung an die Website zurück.
- Sitzungen (oder Sessions) ermöglichen es einer Website, Daten über einen Benutzer vorübergehend auf dem Server zu speichern. Wenn ein Besucher eine andere Seite innerhalb der Website anfordert, kann der PHP-Interpreter auf die Daten aus der Sitzung des Benutzers zugreifen.

Cookies und Sessions speichern kleine Datenmengen vorübergehend, eine langfristige Speicherung ist jedoch nicht gesichert, da die Nutzer Cookies löschen können (oder mit einem anderen Browser auf die Website zugreifen, der das Cookie nicht gesetzt hat), und Sessions sind nur für die Dauer eines Besuchs ausgelegt (sie speichern keine Daten zwischen den Besuchen einer Website).

Wenn personenbezogene Daten über einen längeren Zeitraum gespeichert werden müssen, werden sie in einer Datenbank abgelegt. Wie das funktioniert, erfahren Sie in Kapitel 13. Dazu müssen Sie wissen, wie Cookies und Sessions funktionieren.



WAS SIND COOKIES?

Eine Website kann einen Browser anweisen, Daten über den Benutzer in einer als Cookie bezeichneten Textdatei zu speichern. Bei jeder Anforderung einer anderen Seite dieser Website sendet der Browser dann auch die im Cookie enthaltenen Daten zurück an den Server.

WAS IST EIN COOKIE?

Eine Website kann einen Browser zur Erstellung eines Cookies auffordern, also einer Textdatei, die im Browser gespeichert wird.

Jedes Cookie hat einen Namen, der die Art der darin gespeicherten Informationen beschreiben sollte. Der Cookie-Name bleibt für jeden Besucher gleich.

Der im Cookie eines jeden Benutzers gespeicherte Wert kann sich ändern. Ein Cookie ist also wie eine Variable, die in einer Textdatei im Browser gespeichert wird.

WER HAT ZUGRIFF AUF EIN COOKIE?

Browser senden nur dann in einem Cookie gespeicherte Daten an einen Server, wenn sie eine Seite von der derselben Domäne anfordern, die auch das Cookie erstellt hat. Wenn z.B. google.com ein Cookie erstellt, wird dieses nur gesendet, wenn der Browser Seiten von google.com anfordert. Es würde niemals an facebook.com gesendet werden.

JavaScript kann auch auf Cookie-Daten zugreifen, wenn sie von derselben Domäne gesendet wurden, die den Cookie erstellt hat.

COOKIES ERSTELLEN

Bei der Anforderung einer Webseite durch einen Browser kann die Webseite zusammen mit der Seite einen zusätzlichen HTTP-Header an den Browser rücksenden.

Dieser HTTP-Header teilt dem Browser den Namen des zu erstellenden Cookies und den Wert mit, der in dem Cookie gespeichert werden soll.

Der im Cookie gespeicherte Wert ist ein Text, der nicht länger als 4.096 Zeichen sein sollte. Eine Website kann auch mehrere Cookies erstellen.

COOKIES SIND AN EINEN BROWSER GEBUNDEN

- Weil die Cookies vom Browser erstellt und gespeichert werden:
- Sind mehrere Browser auf einem Gerät installiert, wird das Cookie nur von dem Browser gesendet, der es gespeichert hat (und nicht von sämtlichen Browsern, die auf dem Gerät installiert sind).
- Wenn ein Benutzer ein neues Gerät bekommt, verfügt dieses nicht über das Cookie, das es an den Server senden könnte.

DATEN AUS COOKIES ABRUFEN

Wenn der Browser eine andere Seite von der Website anfordert, die das Cookie erstellt hat, übermittelt er zusammen mit der Seitenanforderung den Namen des Cookies und den darin gespeicherten Wert an den Server.

Der PHP-Interpreter fügt dann die Cookie-Daten in das superglobale Array `$_COOKIE` ein, damit der PHP-Code auf der Seite darauf zugreifen kann. Der Schlüssel ist der Name des Cookies und dessen Wert der im Cookie hinterlegte Wert.

WIE LANGE HALTEN COOKIES?

Der Server kann ein Datum und eine Uhrzeit als Ablaufdatum für ein Cookie festlegen. Ab diesem Zeitpunkt sollte der Browser die Daten im Cookie nicht mehr an den Server übermitteln. Wird kein Ablaufdatum angegeben, sendet der Browser das Cookie nicht mehr an den Server, sobald der Nutzer seinen Browser schließt.

Anwender können Cookies auch ablehnen oder löschen, sodass eine Website auch ohne sie funktionieren sollte.

ERSTE SEITENANFRAGE

- Browser fordert per HTTP-Request eine Seite an

→ http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

REQUEST: page.php

- Server sendet die angeforderte Seite zurück
- Fügt einen HTTP-Header hinzu, der dem Browser den Namen und den Wert eines zu erstellenden Cookies mitteilt

RESPONSE: page.php

HEADER: counter = 1

- Der Browser zeigt die Seite an
- Erstellt ein Cookie mit den im HTTP-Header enthaltenen Daten

http://eg.link/page.php

1

COOKIE: counter = 1

Ein Cookie sollte nicht zum Speichern sensibler Daten (z.B. E-Mail-Adressen oder Kreditkartennummern) verwendet werden, da die Daten im Cookie in den Entwickler-Tools eines Browsers eingesehen werden können und zwischen Browser und Server im Klartext übermittelt werden.

NACHFOLGENDE SEITENANFRAGEN

- Browser fordert per HTTP-Request eine Seite an
- Sendet HTTP-Header mit Name und Wert des Cookies

→ http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

1

REQUEST: page.php

HEADER: counter = 1

- Server fügt Cookie-Daten zu \$_COOKIE hinzu
- Erstellt die Seite mit den Daten in \$_COOKIE
- Sendet die angeforderte Seite zurück
- Kann den im Cookie gespeicherten Wert aktualisieren

RESPONSE: page.php

HEADER: counter = 2

- Browser zeigt die mithilfe seiner Cookie-Daten erstellte Seite an
- Aktualisiert das Cookie anhand der Daten im HTTP-Header

http://eg.link/page.php

2

COOKIE: counter = 2

Um zu verhindern, dass jemand die HTTP-Header bei der Übermittlung zwischen Browser und Server ausliest, sollten Sie Ihre Website mit HTTPS und nicht mit HTTP betreiben (siehe Seite 184). Dadurch werden die Informationen in den Headern verschlüsselt.

COOKIES ERSTELLEN UND DARAUF ZUGREIFEN

Die in PHP integrierte Funktion `setcookie()` dient zur Erstellung eines Cookies. Für den Zugriff auf Cookies können Sie das superglobale Array `$_COOKIE` oder die Funktionen `filter_input()` und `filter_input_array()` verwenden.

Die PHP-Funktion `setcookie()` erstellt einen HTTP-Header, der gemeinsam mit der Webseite versendet wird und den Browser anweist, ein Cookie zu erstellen. Mit dieser Funktion können Sie einen Namen und einen Wert für das Cookie festlegen.

Da die Funktion einen HTTP-Header erzeugt, muss sie vor der Übertragung des Inhalts an den Browser verwendet werden (so wie für `header()` auf Seite 226 gezeigt). Auch ein Leerzeichen vor dem öffnenden `<?php`-Tag wird als Inhalt behandelt.

Wenn Sie für ein Cookie kein Ablaufdatum festlegen, sendet der Browser die Cookie-Daten nicht mehr an den Server, sobald der Browser geschlossen wird. Wie Sie ein Ablaufdatum für ein Cookie festlegen können, erfahren Sie auf Seite 336.

```
setcookie($name, $value);
```

Wenn der Browser das Cookie gespeichert hat, werden dessen Name und der Wert mit dem Request an den Server gesendet, sobald der Browser eine andere Seite der Website anfordert. Wenn der PHP-Interpreter die Anfrage erhält, fügt er die Cookie-Daten in das superglobale Array `$_COOKIE` ein.

Für jedes Cookie wird ein neues Element in das Array eingefügt:

- Der Schlüssel ist der Name des Cookies
- Der Wert ist der im Cookie (in Form einer Zeichenkette) hinterlegte Wert

Diese Daten werden häufig abgerufen und gemeinsam in einer Variablen gespeichert.

Wenn der Code versucht, auf einen nicht im Array `$_COOKIE` vorhandenen Schlüssel zuzugreifen, wird ein Fehler erzeugt. Um dies zu verhindern, kann der Null-Koaleszenz-Operator verwendet werden, um zu prüfen, ob der Schlüssel im Array existiert. Ist dies der Fall, wird der Wert aus dem Cookie in der Variablen gespeichert. Wenn nicht, speichert die Variable den Wert `null`.

```
$preference = $_COOKIE['name'] ?? null;
```

Die PHP-Funktionen `filter_input()` und `filter_input_array()` (Seite 268) können ebenfalls Cookie-Daten sammeln. Der Eingabetyp sollte auf `INPUT_COOKIE` gesetzt werden.

Der zweite Parameter ist der Name des Cookies. Der dritte und der vierte Parameter sind optional; sie geben die ID des einzusetzenden Filters und etwaige Filteroptionen an.

Falls kein Cookie gesendet wurde, löst die Funktion keinen Fehler aus. Wenn Integer-, Float- oder Boolean-Filter verwendet werden, konvertieren sie den Wert in den entsprechenden Datentyp.

```
$preference = filter_input(INPUT_COOKIE, $name[, $filter[, $options]]);
```

PHP

section_b/c09/cookies.php

```
<?php  
① $counter = $_COOKIE['counter'] ?? 0; // Daten abrufen  
② $counter = $counter + 1;           // Zähler +1 erhöhen  
③ setcookie('counter', $counter);   // Cookie aktualisieren  
  
④ $message = 'Page views: ' . $counter; // Nachricht  
?>  
<?php include 'includes/header.php'; ?>  
  
<h1>Welcome</h1>  
⑤ <p><?= $message ?></p>  
<p><a href="sessions.php">Refresh this page</a> to see  
the page views increase.</p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



In diesem Beispiel dient ein Cookie dazu, die Anzahl der Seiten zu zählen, die ein Besucher aufgerufen hat.

1. Die Variable `$counter` speichert die Anzahl der Seiten, die sich der Besucher angesehen hat. Wenn der Browser Daten aus dem Cookie `counter` an den Server sendet, speichert `$counter` diesen Wert. Ist dies nicht der Fall, wird stattdessen mit dem Null-Koaleszenz-Operator der Wert 0 gespeichert.
2. Zu dem Wert in `$counter` wird 1 addiert, da der Besucher sich gerade eine Seite angesehen hat.
3. Die Funktion `setcookie()` weist den Browser an, das Cookie `counter` zu erstellen oder zu aktualisieren und den Wert aus `$counter` in diesem Cookie zu speichern.
4. Die Variable `$message` speichert eine Nachricht über die vom Besucher betrachtete Seitenanzahl.
5. Die Nachricht wird angezeigt.

Probieren Sie es: Wenn Sie die Seite einmal angesehen haben, aktualisieren Sie sie und beobachten, wie der Zähler ansteigt.

Probieren Sie es: Speichern Sie Ihren Namen im Cookie Name und zeigen Sie ihn nach dem Seitenaufruf an.

COOKIES ABSICHERN

`setcookie()` hat Parameter, um den Umgang von Browsern mit Cookies zu regeln. Über Cookies empfangenen Daten sollten Sie zudem validieren und `htmlspecialchars()` zur Ausgabe der Inhalte auf einer Seite verwenden.

Um einen in einem Cookie gespeicherten Wert zu aktualisieren, rufen Sie `setcookie()` erneut mit einem neuen Wert für das Cookie auf. Um zu unterbinden, dass der Browser ein Cookie sendet, rufen Sie `setcookie()` erneut auf. Setzen Sie als Wert eine leere Zeichenkette und als Ablaufdatum einen Zeitpunkt in der Vergangenheit. Wenn Sie den Wert oder das Ablaufdatum eines Cookies aktualisieren, müssen die letzten vier Argumente dieselben Werte verwenden, die bei der Erstellung des Cookies genutzt wurden.

`setcookie($name[, $value, $expire, $path, $domain, $secure, $httponly])`

PARAMETER BESCHREIBUNG

<code>\$name</code>	Der Name des Cookies.																		
<code>\$value</code>	Der Wert, den das Cookie enthalten soll (wird als String behandelt - Cookies speichern keine Datentypen).																		
<code>\$expire</code>	Das Datum und die Uhrzeit, ab wann der Browser das Cookie nicht mehr an den Server senden soll (als Unix-Zeitstempel). <table><thead><tr><th>ZEIT-RAUM</th><th>JETZT</th><th>SEK</th><th>MIN</th><th>H</th><th>TAGE</th></tr></thead><tbody><tr><td>1 Tag</td><td>time() + 60 * 60 * 24</td><td></td><td></td><td></td><td></td></tr><tr><td>30 Tage</td><td>time() + 60 * 60 * 24 * 30</td><td></td><td></td><td></td><td></td></tr></tbody></table>	ZEIT-RAUM	JETZT	SEK	MIN	H	TAGE	1 Tag	time() + 60 * 60 * 24					30 Tage	time() + 60 * 60 * 24 * 30				
ZEIT-RAUM	JETZT	SEK	MIN	H	TAGE														
1 Tag	time() + 60 * 60 * 24																		
30 Tage	time() + 60 * 60 * 24 * 30																		
	Um den Zeitstempel zu setzen, nutzen Sie die PHP-Funktion <code>time()</code> und geben den gewünschten Gültigkeitszeitraum für das Cookie an.																		
<code>\$path</code>	Wenn ein Cookie nur für einen Teil der Website benötigt wird, geben Sie die Verzeichnisse an, für die er verwendet werden soll. Standardmäßig entspricht der Pfad dem Wurzelverzeichnis /, umfasst also alle Verzeichnisse. Die Änderung auf /members bedeutet, dass das Cookie nur an Seiten im members-Ordner der Website gesendet wird.																		
<code>\$domain</code>	Wenn das Cookie nur auf einer Subdomain benötigt wird, legen Sie die URL für die Subdomain fest. Standardmäßig wird es an alle Subdomains einer Website gesendet. Wird der Wert auf die Subdomain <code>members.example.org</code> gesetzt, wird das Cookie nur an Dateien in dieser Subdomain gesendet.																		
<code>\$secure</code>	Wird hier der Wert <code>true</code> angegeben, wird das Cookie im Browser erstellt, aber der Browser sendet es nur an den Server zurück, wenn die Seite über eine sichere HTTPS-Verbindung angefordert wird (siehe Seite 184).																		
<code>\$httponly</code>	Wird der Wert <code>true</code> angegeben, wird das Cookie nur an den Server gesendet (JavaScript kann nicht darauf zugreifen).																		

Es gibt die Möglichkeit, mit einem Page-Request HTTP-Header zu übertragen, die Cookies imitieren. Deshalb:

- Der Server sollte Cookie-Daten validieren, bevor er sie weiterverwendet (mit Techniken aus Kapitel 6).
- Wenn ein Cookie-Wert auf einer Seite angezeigt wird, sollte dies mit `htmlspecialchars()` erfolgen, um einer XSS-Attacke vorzubeugen.

COOKIE-OPTIONEN FESTLEGEN

PHP

section_b/c09/cookie-preferences.php

```
<?php  
① $color = $_COOKIE['color'] ?? null;           // Daten abrufen  
② $options = ['light', 'dark',];                // Optionen  
  
③ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn POST  
④     $color = $_POST['color'];                 // Farbe abrufen  
⑤     setcookie('color', $color, time() + 60 * 60,  
     '/'); //, false, true);                   // Cookie setzen  
}  
  
// Wenn color gültige Option, diese verwenden, sonst dark.  
⑥ $scheme = (in_array($color, $options)) ? $color : 'dark';  
?  
⑦ <?php include 'includes/header-style-switcher.php'; ?>  
    <form method="POST" action="cookie-preferences.php">  
        Select color scheme:  
        <select name="color">  
            <option value="dark">Dark</option>  
            <option value="light">Light</option>  
        </select><br>  
        <input type="submit" value="Save">  
    </form>  
<?php include 'includes/footer.php'; ?>
```

PHP

section_b/c09/includes/header-style-switcher.php

```
<body class="<?= htmlspecialchars($scheme) ?>">
```

ERGEBNIS



Quelle: Microsoft Corporation, mit freundlicher Genehmigung

1. Die Variable `$color` speichert den Wert, der für das Cookie `color` gesendet wurde (oder `null`, wenn dieses nicht gesendet wurde).
2. Ein Array speichert die zulässigen Optionen für das Farbschema.
3. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
4. Ist dies der Fall, wird der Wert für das Auswahlfeld `color` in der Variablen `$color` gespeichert. Damit wird der Wert aus Schritt 1 überschrieben.
5. Die Funktion `setcookie()` wird aufgerufen, um das Cookie `color` zu setzen. Sein Wert entspricht der Option, die der Benutzer in der Auswahlbox gewählt hat. Außerdem gilt für dieses Cookie:
 - Es läuft nach einer Stunde ab
 - Es wird an alle Seiten der Webseite gesendet
 - Es wird über HTTP oder HTTPS verschickt
 - Es wird vor JavaScript verborgen
6. Die Bedingung eines ternären Operators prüft, ob der Wert in `$color` im Array `$options` enthalten ist. Ist dies der Fall, wird der Wert in der Variablen `$scheme` gespeichert. Wenn nicht, wird in `$scheme` der Wert `dark` abgelegt.
7. Ein neuer Header wird eingefügt. Der Wert in der Variablen `$color` wird in das `class`-Attribut des `<body>`-Tags geschrieben, um sicherzustellen, dass die CSS-Regeln für die Seite das richtige Farbschema verwenden.

WAS SIND SESSIONS?

Sessions oder Sitzungen speichern Informationen über einen Benutzer und seine Präferenzen auf dem Server. Sie heißen Sitzungen, weil sie die Daten nur für die Dauer eines einzigen Besuchs auf der Website speichern.

WAS IST EINE SESSION?

Zu Beginn einer Sitzung erstellt der PHP-Interpreter drei Dinge:

- Eine Session-ID, also einen String, der zur Identifizierung eines einzelnen Besuchers dient.
- Eine Session-Datei, also eine Textdatei, die auf dem Server gespeichert wird. Sie enthält Daten über diesen Benutzer. Ihr Dateiname enthält die Sitzungs-ID.
- Ein Session-Cookie, das im Browser gespeichert wird. Sein Name lautet PHPSESSID und sein Wert entspricht der Session-ID des betreffenden Benutzers.

WIE LANGE KÖNNEN SITZUNGEN DAUERN?

Eine funktionsfähige Session benötigt sowohl das Session-Cookie im Browser als auch die Session-Datei auf dem Server.

- Session-Cookies laufen beim Schließen des Browsers ab. Session-Dateien können vom Server gelöscht werden, wenn sie nicht innerhalb eines bestimmten Zeitraums (standardmäßig 24 Minuten) geändert werden.

SITZUNGSDATEN ABRUFEN

Wenn ein Browser über ein Session-Cookie verfügt, wird dieses jedes Mal an den Server gesendet, wenn der Benutzer eine andere Seite von dieser Website anfordert. Die Session-ID wird zur Identifizierung des Benutzers verwendet, sodass der Server:

- Die Session-Datei finden kann, deren Dateiname die im Cookie übertragene Session-ID enthält.
- Daten aus der Session-Datei nehmen und in das superglobale Array `$_SESSION` einfügen kann, um der Seite den Zugriff darauf zu ermöglichen.

WIE WERDEN SITZUNGEN GESTARTET?

Wenn eine Website Sitzungen verwendet, sollte jede Seite die PHP-Funktion `session_start()` aufrufen. Wenn diese Funktion aufgerufen wird und der Browser, der die Seite anfordert, kein Session-Cookie gesendet hat oder wenn keine passende Session-Datei gefunden werden kann, startet der PHP-Interpreter für diesen Benutzer automatisch eine neue Sitzung.

SITZUNGSDATEN SPEICHERN

Sobald eine Sitzung angelegt wurde, können neue Daten durch Hinzufügen zum superglobalen Array `$_SESSION` in der Sitzung dieses Benutzers gespeichert werden.

Wenn die Ausführung einer Seite beendet wurde, nimmt der PHP-Interpreter alle Daten aus dem superglobalen Array `$_SESSION` und speichert sie in der Session-Datei des Benutzers. Beim Speichern von Daten wird zugleich auch die Zeit der letzten Änderung aktualisiert, und der PHP-Interpreter kann darüber feststellen, ob eine Session kürzlich benutzt wurde.

WEITERE NUTZUNGSMÖGLICHKEITEN FÜR SESSIONS

Anstatt Session-Cookies zu verwenden, ist es auch möglich, zu URLs eine Session-ID hinzuzufügen, was jedoch weniger sicher ist. Sitzungsdaten können zudem auch in einer Datenbank gespeichert werden, aber dieses Thema würde den Rahmen dieses Buchs sprengen. (Dies wird in der Regel nur auf Websites mit sehr hohem Traffic verwendet, die mehrere Server zur Bewältigung der Last benötigen.)

ERSTE SEITENANFRAGE

- Browser fordert per HTTP-Request eine Seite an

http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

REQUEST: page.php

Auf dem Server ruft die PHP-Seite `session_start()` auf. Der Browser hat kein Session-Cookie gesendet, also erzeugt sie:

- eine Session-ID für diesen Benutzer
- eine Session-Datei mit den Daten des Benutzers (der Dateiname enthält die Session-ID)

Die Seite fügt Daten in das superglobale Array `$_SESSION` ein; wenn die Ausführung der Seite beendet ist, werden die Werte aus diesem Array in die Session-Datei geschrieben, die sie für diesen Benutzer erstellt hat.

- Der Server sendet die angeforderte Seite zurück
- Sendet einen HTTP-Header, der ein Session-Cookie mit der Session-ID erstellt

RESPONSE: page.php

HEADER: PHPSESSID = 1234567

- Der Browser zeigt die Seite an
- Erzeugt Session-Cookie mit Session-ID

http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

1

COOKIE: PHPSESSID = 1234567

NACHFOLGENDE SEITENANFRAGEN

- Browser fordert per HTTP-Request eine Seite an
- Sendet HTTP-Header mit der Session-ID

http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

1

REQUEST: page.php

HEADER: PHPSESSID = 1234567

Auf dem Server ruft die PHP-Seite `session_start()` auf. Der PHP-Interpreter findet die Session-Datei mit der im Session-Cookie angegebenen Session-ID und:

- fügt Daten aus der Session-Datei in das superglobale Array `$_SESSION` ein, damit die Seite diese Daten verwenden kann
- erstellt eine Seite mit Daten aus dem Array
- kann Daten im Array aktualisieren

Wenn die Ausführung der Seite beendet ist, werden die Werte im superglobalen Array `$_SESSION` in der Session-Datei gespeichert. Dabei wird auch der Zeitpunkt der letzten Änderung in der Session-Datei aktualisiert.

- Der Server sendet die angeforderte Seite zurück

RESPONSE: page.php

http://eg.link/page.php

Quelle: Microsoft Corporation,
mit freundlicher Genehmigung

2

COOKIE: PHPSESSID = 1234567

SITZUNGEN ERSTELLEN UND DARAUF ZUGREIFEN

Jede Seite einer Website, die Sitzungen verwendet, sollte `session_start()` aufrufen. Wenn es für den Benutzer keine offene Sitzung gibt, wird eine gestartet; gibt es eine, werden die Sitzungsdaten abgerufen und in das superglobale Array `$_SESSION` eingefügt.

Wenn ein Besucher eine Seite, die `session_start()` aufruft, erstmalig anfordert, werden eine neue Session-ID, ein Session-Cookie und eine Session-Datei erstellt.

Die Funktion muss vor der Übertragung eines Inhalts an den Browser aufgerufen werden, da sie einen HTTP-Header zur Erstellung des Session-Cookies sendet.

Der Aufruf muss auch erfolgen, bevor die Seite versucht, Session-Daten abzurufen, da die Daten aus der Session-Datei in das superglobale Array `$_SESSION` übertragen werden.

`session_start();`

Wenn Sie Daten in das superglobale Array `$_SESSION` schreiben, fügt der PHP-Interpreter Die Daten nach Abschluss der Seitenausführung in die Session-Datei dieses Benutzers ein.

Die Syntax zum Hinzufügen von Daten zum Array ist analog zu jedem assoziativen Array. Der Schlüssel sollte die vom Element zu speichernden Daten beschreiben.

Der Wert zu jedem Schlüssel kann ein skalarer Wert (String, Zahl oder boolescher Wert), ein Array oder ein Objekt sein. Dieser Datentyp wird mit gespeichert (anders als bei Cookies, die nur Zeichenketten aufnehmen).

```
$_SESSION['name'] = 'Ivy';
$_SESSION['age'] = 27;
```

Bei der Erfassung von Daten aus dem superglobalen Array `$_SESSION` verwenden Sie

den Null-Koaleszenz-Operator für den Fall, dass Werte fehlen,

oder nutzen die PHP-Filterfunktionen mit dem Eingabetyp `INPUT_SESSION`.

```
$name = $_SESSION['username'] ?? null;
$age = $_SESSION['age'] ?? null;
```

FUNKTION	BESCHREIBUNG
<code>session_start()</code>	Neue Sitzung erstellen oder Daten aus einer bestehenden Sitzung abrufen.
<code>session_set_cookie_params()</code>	Einstellungen zum Anlegen des Session-Cookies (dieselben Parameter wie auf Seite 336).
<code>session_get_cookie_params()</code>	Liefert ein Array mit den Argumenten zurück, die zum Setzen des Cookies verwendet werden.
<code>session_regenerate_id()</code>	Erzeugt eine neue Session-ID und aktualisiert die Session-Datei und das Cookie..
<code>session_destroy()</code>	Löscht die Session-Datei vom Server.

DATEN IN SITZUNGEN SPEICHERN UND DARAUF ZUGREIFEN

PHP

section_b/c09/sessions.php

```
<?php  
① session_start(); // anlegen/wiederaufnehmen  
② $counter = $_SESSION['counter'] ?? 0; // Daten abrufen  
③ $counter = $counter + 1; // Zähler +1 erhöhen  
④ $_SESSION['counter'] = $counter; // Session aktualisieren  
  
⑤ $message = 'Page views: ' . $counter; // Nachricht  
?>  
<?php include 'includes/header.php'; ?>  
  
<h1>Welcome</h1>  
⑥ <p><?= $message ?></p>  
<p><a href="sessions.php">Refresh this page</a>  
a to see the page views increase.</p>  
  
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



Probieren Sie es: Nachdem Sie die Seite einmal angesehen haben, aktualisieren Sie sie und beobachten dabei, wie der Zähler hochzählt.

Probieren Sie es: Speichern Sie Ihren Namen im superglobalen Array `$_SESSION` und geben Sie ihn auf der Seite aus.

Dieses Beispiel erfüllt die gleiche Aufgabe wie jenes auf Seite 335, aber der Zähler wird hier in einer Sitzung gespeichert.

1. Wenn die PHP-Funktion `session_start()` aufgerufen wird, versucht der PHP-Interpreter, die Daten aus der Session-Datei abzurufen und im superglobalen Array `$_SESSION` zu speichern. Wenn dies nicht gelingt, wird eine neue Session für diesen Besucher erstellt.
2. Wenn der Schlüssel `counter` im superglobalen Array `$_SESSION` einen Wert hat, wird er in der Variablen `$counter` gespeichert. Andernfalls enthält `$counter` den Wert 0.
3. Der Besucher hat gerade eine Seite betrachtet, also wird der Zähler um 1 erhöht.
4. Der Wert für den Schlüssel `counter` im superglobalen Array `$_SESSION` wird aktualisiert.
5. Die Variable `$message` speichert eine Nachricht über die Anzahl der Seiten, die der Besucher angesehen hat.
6. Die Nachricht wird angezeigt. Nachdem die Seite ausgeführt wurde, speichert PHP die Daten aus dem superglobalen Array `$_SESSION` in der Session-Datei für diesen Benutzer. Beim Speichern der Daten wird zugleich der Zeitpunkt der letzten Änderung der Session-Datei auf dem Server aktualisiert, sodass die Session-Daten länger gültig sind.

LEBENSDAUER EINER SITZUNG

Browser löschen Session-Cookies beim Schließen des Browserfensters.
Server löschen Session-Dateien, wenn der Garbage-Collection-Prozess läuft. Demzufolge können Sitzungen länger dauern, als Sie es erwarten.

Falls Sie dies noch nicht getan haben, öffnen Sie das vorherige Beispiel in Ihrem Browser. Öffnen Sie dann:

- die Entwicklungswerzeuge des Browsers zur Anzeige von Cookies
- den Ordner, in dem der Webserver Session-Dateien speichert

Hilfe hierzu finden Sie jeweils unter:

<http://notes.re/php/session-locations>

Im Browser sollten Sie das Cookie PHPSESSID sehen. Der Wert dieses Cookies entspricht der Session-ID.

The screenshot shows the Chrome DevTools interface with the 'Network' tab selected. In the left sidebar, under 'Cookies', there is a list for the domain 'https://phpbook'. One cookie is visible: 'PHPSESSID' with the value '2rnkbsj0qahv7e920n3othilg8'. This cookie is highlighted in blue. The right side of the screen shows a table of all cookies for this domain, with columns for Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, SameSite, and Priority. The 'PHPSESSID' row is also highlighted in blue in this table.

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

In dem Ordner, in dem der Webserver die Session-Dateien speichert, sollten Sie einen Dateinamen sehen, der die Session-ID enthält. Notieren Sie das Datum und die Uhrzeit, zu der die Session-Datei zuletzt geändert wurde, und aktualisieren Sie dann das Browserfenster, das das vorherige Beispiel anzeigt; die Uhrzeit der letzten Änderung wird dann aktualisiert. Wenn Sie das Beispiel in einem anderen Webbrowser öffnen (probieren Sie es z.B. mit Chrome und Firefox), wird eine neue Sitzung erstellt, da die Session an den Browser gebunden ist.

Wenn eine Seite `session_start()` aufruft und der PHP-Interpreter kein Session-Cookie erhält oder keine passende Session-Datei für dieses Session-Cookie finden kann, wird eine neue Session erstellt.

Wenn die Ausführung einer Seite, die `session_start()` aufgerufen hat, beendet wird, speichert sie die Daten aus dem superglobalen Array `$_SESSION` in der Session-Datei. Hierdurch wird auch die Zeit der letzten Änderung in der Session-Datei aktualisiert.

sess_0lucsl...8fusgrc94q
sess_1rp24...sdccjceo1nb
sess_2rnkb...20n3othilg8
sess_r565c...book7hubjh
sess_uib716...p216vrvm7ni

sess_2rnkbsj0qahv7e920n3othilg8
TextEdit.app Document - 12 bytes
Information
Created Today, 15:34
Modified Today, 15:39

Quelle: Microsoft Corporation, mit freundlicher Genehmigung

Anhand des Zeitpunkts der letzten Änderung bestimmt der PHP-Interpreter, wann er die Session-Datei löschen kann (was die Sitzung dann zugleich beendet).

Wenn eine Website Sitzungen verwendet, ist es daher wichtig, auf jeder Seite `session_start()` aufzurufen. Andernfalls könnte die Sitzung vorzeitig ablaufen, während der Benutzer noch auf der Website surft.

Der Webserver führt den Prozess Garbage Collection aus. Dieser löscht Session-Dateien, deren letztes Änderungsdatum ein bestimmtes Alter überschreitet (der Standardwert liegt bei 24 Minuten). Sobald eine Session-Datei gelöscht wurde, ist die Sitzung beendet, denn selbst wenn ein Browser das Session-Cookie gesendet hätte, könnte die Datei mit den Sitzungsdaten nicht mehr gefunden werden.

Die Überprüfung des letzten Zugriffs auf die einzelnen Session-Dateien und das Löschen der alten Session-Dateien beansprucht Server-Ressourcen, weshalb die Server dies möglichst nicht allzu häufig durchführen. Die Zeitabstände, in denen die Garbage Collection durchgeführt wird, hängen von der Zugriffszahl auf die Sitzungen ab. Daher kann es auf sehr ruhigen Websites vorkommen, dass die Garbage Collection stunden- oder sogar tagelang nicht erfolgt.

1. Alle Daten aus der Session-Datei entfernen, indem dem superglobalen Array `$_SESSION` ein leeres Array zugewiesen wird.

```
$_SESSION = [];
```

2. In Schritt 3 dient `setcookie()` zur Aktualisierung des Session-Cookies. Dabei müssen die Argumente für die Parameter `path`, `domain`, `secure` und `httponly` dieselben Werte haben, die bei der Erstellung des Cookies verwendet wurden.

```
$params = session_get_cookie_params();
```

3. Mit der PHP-Funktion `setcookie()` (siehe Seite 334) wird das Session-Cookie aktualisiert. Als `value`-Parameter wird eine leere Zeichenkette verwendet; dies löscht die Session-ID aus dem Session-Cookie.

```
setcookie('PHPSESSID', '', time() - 3600, $params['path'],  
$params['domain'], $params['secure'], $params['httponly']);
```

4. Die PHP-Funktion `session_destroy()` aufrufen, um den PHP-Interpreter anzuweisen, die Session-Datei zu löschen.

```
session_destroy();
```

Wie Sie im nächsten Beispiel erkennen, dienen Sitzungen häufig als Erinnerung daran, dass ein Benutzer bei einer Website angemeldet ist. In solchen Fällen sollte den Benutzern die Möglichkeit gegeben werden, sich abzumelden.

Wenn ein Benutzer sein Browserfenster nicht schließt (sodass der Browser immer noch ein Session-Cookie sendet) und der Server ruhig ist (sodass er keine Garbage Collection durchgeführt), kann die Sitzung länger dauern als ursprünglich vorgesehen.

Das ist ein großes Problem, wenn sich mehrere Benutzer einen Computer teilen, denn wenn sich ein Benutzer nicht abmeldet, könnte jemand anderes die Website mit seinem Konto besuchen.

Zum Beenden einer Sitzung sind die folgenden vier Schritte erforderlich.

Dies verhindert auch, dass nachfolgender Code auf derselben Seite auf diese Werte zugreifen kann.

Die PHP-Funktion `session_get_cookie_params()` gibt die Werte zurück, die bei der Erstellung des Session-Cookies für diese Parameter verwendet wurden. Die zurückgelieferten Werte werden als Array in der Variablen `$params` gespeichert.

Für den Parameter `expires` wird ein in der Vergangenheit liegendes Datum gewählt. Dies verhindert, dass der Browser das Cookie bei weiteren Seitenanfragen an den Server sendet. Alle anderen Parameter werden anhand der in Schritt 2 ermittelten Werte festgelegt und als Array in der Variablen `$params` gespeichert.

Der PHP-Interpreter löscht die Datei sofort, anstatt auf die Garbage Collection zu warten.

EIN EINFACHES LOGIN-SYSTEM

Websites verlangen oft, dass sich die Benutzer anmelden, bevor sie bestimmte Seiten aufrufen können. In diesem Beispiel müssen sich die Benutzer anmelden, um die Seite My Account anzuzeigen. Wenn sich ein Benutzer anmeldet:

- merkt sich die Sitzung, dass er sich angemeldet hat
- kann er die Seite My Account aufrufen
- ändert sich der Verknüpfungstext für den letzten Link in der Navigationsleiste von »Log in« in »Logout«

Hinweis: Dieses Beispiel zeigt nur, wie Sitzungen verwendet werden, um sich zu merken, wann sich ein Benutzer angemeldet hat. In Kapitel 16 stellen wir ein vollständiges Anmeldesystem vor, bei dem jedes Mitglied seine eigenen Anmelddaten hat (die in einer Datenbank gespeichert werden).

Wenn eine Website Sitzungen verwendet, sollte jede Unterseite `session_start()` aufrufen, bevor sie irgendwelche Inhalte an den Browser sendet. Damit wird sichergestellt, dass jeder Benutzer eine eigene Session hat und dass der Zeitpunkt der letzten Änderung der Session-Datei bei jedem neuen Seitenaufruf aktualisiert wird.

In diesem Beispiel bindet jede Seite die Datei `sessions.php` (siehe rechte Seite) ein. Diese ruft `session_start()` auf und enthält den ganzen sitzungsbezogenen Code.

1. Die Funktion `session_start()` weist den PHP-Interpreter an, die Daten aus der Session-Datei des Besuchers in das superglobale Array `$_SESSION` zu übernehmen oder eine neue Sitzung zu erstellen, falls dies nicht gelingt.
2. Wenn das superglobale Array `$_SESSION` die Benutzeranmeldung aufgezeichnet hat, erhält die Variable `$logged_in` den Wert `true`; andernfalls weist der Null-Koalescenz-Operator ihr den Wert `false` zu.
3. Die Variablen `$email` und `$password` enthalten die für die Anmeldung erforderlichen Benutzerangaben.

Die Datei enthält dann drei Funktionsdefinitionen:

4. Die Anmeldeseite ruft die Funktion `login()` auf, wenn der Benutzer die richtige E-Mail und das richtige Passwort eingibt.
5. Wenn sich ein Benutzer anmeldet, ist es sinnvoll, seine Session-ID zurückzusetzen. Die PHP-Funktion `session_regenerate_id()` erzeugt eine neue Session-ID und aktualisiert die Session-Datei und das Cookie mit dieser neuen Session-ID. (Das Argument `true` weist den PHP-Interpreter an, alle bereits in der Session vorhandenen Daten zu löschen).
6. Der Schlüssel `logged_in` wird der Session hinzugefügt. Er hat den Wert `true` und zeigt an, dass der Besucher angemeldet ist.
7. Mit der Funktion `logout()` wird die Sitzung beendet.
8. Dem superglobalen Array `$_SESSION` wird ein leeres Array zugewiesen. Damit werden die Daten aus der Session-Datei gelöscht, und der Rest der Seite greift nicht mehr darauf zurück.
9. Das Session-Cookie wird aktualisiert; die Session-ID wird durch eine leere Zeichenkette ersetzt, und das Ablaufdatum wird in der Vergangenheit angesiedelt (sodass der Browser das Cookie nicht mehr sendet).
10. Die Session-Datei auf dem Server wird gelöscht.
11. Die Funktion `require_login()` kann von jeder Seite aufgerufen werden, die eine Anmeldung des Besuchers erfordert.
12. Eine `if`-Anweisung prüft, ob die Variable `$logged_in` `false` enthält. Ist dies der Fall, hat sich der Benutzer entweder nicht eingeloggt oder die Sitzung wurde beendet.
13. Der Nutzer wird zur Anmeldeseite umgeleitet.
14. Durch den `exit`-Befehl wird kein weiterer Code mehr ausgeführt.

ERGEBNIS



PHP

```
<?php  
① session_start();  
② $logged_in = $_SESSION['logged_in'] ?? false;  
  
③ [ $email      = 'ivy@eg.link';  
  $password   = 'password';  
  
④ function login()  
{  
⑤     session_regenerate_id(true);  
⑥     $_SESSION['logged_in'] = true;  
}  
  
⑦ function logout()  
{  
⑧     $_SESSION = [];  
  
⑨     [ $params = session_get_cookie_params();           // Session-Cookie-Parameter abrufen  
      setcookie('PHPSESSID', '', time() - 3600, $params['path'], $params['domain'],  
                $params['secure'], $params['httponly']);    // Session-Cookie löschen  
}  
  
⑩    session_destroy();                                // Session-Datei löschen  
}  
  
⑪ function require_login($logged_in)  
{  
⑫     if ($logged_in == false) {  
⑬         header('Location: login.php');  
⑭         exit;  
}  
}
```

section_b/c09/includes/sessions.php

SO STELLEN SIE SICHER, DASS SICH BENUTZER ZUM BETRACHTEN VON SEITEN ANMELDEN

Die Funktion `require_login()` sollte zu Beginn jeder Seite aufgerufen werden, auf der sich Besucher anmelden müssen. In diesem Beispiel müssen die Besucher eingeloggt sein, um die Seite `account.php` anzuzeigen.

1. Die `sessions.php` include file is included.
2. Die in `sessions.php` definierte Funktion `require_login()` prüft, ob der Benutzer eingeloggt ist. Wenn er:
 - eingeloggt ist, wird der Rest der Seite angezeigt
 - nicht eingeloggt ist, wird er zu `login.php` weitergeleitet

Ihr einziges Argument ist die Variable `$logged_in`, die in Schritt 2 von `sessions.php` deklariert wurde.

3. Eine neue Header-Datei (siehe dritter Code-Kasten) wird eingebunden.
4. Als Nächstes sehen Sie die Seite `login.php`. Sie beginnt mit dem Einbinden der Datei `sessions.php`.
5. Eine `if`-Anweisung überprüft den Wert der Variablen `$logged_in` (die in `sessions.php` erstellt wurde), um festzustellen, ob bereits jemand eingeloggt ist.
6. Wenn sich bereits jemand angemeldet hat, wird die Person auf `account.php` weitergeleitet, da sie sich nicht mehr anmelden muss (möglicherweise ist sie durch Anklicken eines Links oder die Zurück-Schaltfläche des Browsers auf die Seite gelangt).
7. Der `exit`-Befehl verhindert, dass der verbleibende Code der Seite ausgeführt wird.
8. Wenn die Seite noch läuft, prüft die Datei, ob der Benutzer das Formular abgeschickt hat (siehe unten auf der Seite).
9. Wenn ja, werden die Werte, die der Benutzer für die Formularelemente `email` und `password` eingegeben hat, abgerufen und in den Variablen `$user_email` und `$user_password` gespeichert.

10. Eine `if`-Anweisung überprüft, ob die Benutzereingaben für E-Mail-Adresse und Passwort mit den in der Datei `sessions.php` in den Variablen `$email` und `$password` gespeicherten Werten übereinstimmen (siehe Schritt 3 auf der vorherigen Seite).

11. Wenn sie übereinstimmen, hat der Benutzer die richtigen Angaben gemacht, und die in `sessions.php` definierte Funktion `login()` wird aufgerufen. Sie generiert eine neue Session-ID und fügt den Schlüssel `logged_in` zusammen mit dem Wert `true` in das superglobale Array `$_SESSION` ein, um anzuzeigen, dass der Benutzer sich angemeldet hat.

12. Der Benutzer wird dann auf die Seite `account.php` weitergeleitet, und der `exit`-Befehl stoppt die weitere Codeausführung.

13. Wenn das Formular nicht abgeschickt wurde oder die Anmeldedaten falsch waren, wird der Kopfbereich für dieses Beispiel eingefügt.

14. Das Anmeldeformular hat zwei Eingabefelder, eines für die E-Mail-Adresse und eines für das Passwort.

15. Im neuen Kopfbereich überprüft die Navigationsleiste, ob der Benutzer eingeloggt ist. Wenn ja, zeigt sie einen Link zur Abmeldeseite an. Wenn nicht, wird ein Link zur Anmeldeseite angezeigt.

Hinweis: Die Session-ID wird bei jeder Anfrage in den HTTP-Header mitgesendet. Wenn jemand die Session-ID in die Hände bekäme, könnte er eine HTTP-Anfrage erstellen und die Session-ID zu dieser Anfrage hinzufügen und sich als der Benutzer ausgeben, der die Sitzung erstellt hat. Diese Angriffsform wird als Session-Hijacking bezeichnet.

Um Session-Hijacking zu verhindern, sollten alle Zugriffe auf Seiten, die Sessions verwenden, nur über eine HTTPS-Verbindung erfolgen, da hierbei alle Daten verschlüsselt werden (einschließlich der Header mit der Session-ID).

Für dieses Beispiel ist es nicht erforderlich, ein SSL-Zertifikat zu installieren, aber für jede produktive Website sollte dies obligatorisch sein.

```
<?php
① include 'includes/sessions.php';           // Datei sessions.php einbinden
② require_login($logged_in);                 // Nutzer umleiten, falls nicht eingeloggt
?>
③ <?php include 'includes/header-member.php'; ?> ...
```

```
<?php
④ include 'includes/sessions.php';

⑤ if ($logged_in) {                         // wenn bereits eingeloggt
⑥     header('Location: account.php');       // zur Kontoseite weiterleiten
⑦     exit;                                  // weitere Codeausführung stoppen
}

⑧ if($_SERVER['REQUEST_METHOD'] == 'POST') {   // wenn Formular übermittelt
⑨ [   $user_email    = $_POST['email'];      // vom Nutzer übermittelte E-Mail-Adresse
⑩   $user_password = $_POST['password'];      // vom Nutzer übermitteltes Passwort

⑪   if ($user_email == $email and $user_password == $password) { // wenn Anmelddaten stimmen
⑫     login();                                // Login-Funktion aufrufen
⑬     header('Location: account.php');        // zur Kontoseite weiterleiten
⑭     exit;                                  // weitere Codeausführung stoppen
}
?
⑯ <?php include 'includes/header-member.php'; ?>
<h1>Login</h1>
<form method="POST" action="login.php">
⑯ [ Email: <input type="email" name="email"><br>
  Password: <input type="password" name="password"><br>
  <input type="submit" value="Log In">
</form>
<?php include 'includes/footer.php'; ?>
```

```
<a href="home.php">Home</a>
<a href="products.php">Products</a>
<a href="account.php">My Account</a>
⑮ <?= $logged_in ? '<a href="login.php">Log In</a>' : '<a href="logout.php">Log Out</a>' ?>
```

ZUSAMMENFASSUNG

COOKIES & SITZUNGEN

- › Cookies speichern Daten im Browser eines Besuchers.
- › Die in einem Cookie gespeicherten Daten werden der PHP-Seite im superglobalen Array `$_COOKIES` zugänglich gemacht.
- › Sie können das Ablaufdatum von Cookies festlegen (Benutzer können sie aber auch früher löschen).
- › Sessions speichern Daten auf dem Server.
- › Der Zugriff auf Sitzungsdaten und deren Aktualisierung erfolgt über das superglobale Array `$_SESSIONS`.
- › Jede Seite einer Website, die Sitzungen verwendet, sollte zu Beginn die Funktion `session_start()` aufrufen.
- › Session-Dateien speichern Daten während eines einzigen Website-Besuchs (und werden auch nach einem gewissen Zeitraum der Inaktivität gelöscht).
- › Wenn Sie Daten oder persönliche Informationen über einen längeren Zeitraum speichern wollen, verwenden Sie Datenbanken (wie in Kapitel 16 beschrieben).

10

FEHLER- BEHANDLUNG

Wenn der PHP-Interpreter Probleme bei der Ausführung des Codes hat, kann er eine Fehlermeldung erzeugen, die Ihnen bei der Problemsuche hilft.

Wenn Sie eine neue PHP-Seite erstellen, sollten Sie nicht erwarten, dass Sie gleich beim ersten Versuch perfekt läuft; selbst erfahrene Programmierer bekommen beim Testen einer neuen Seite regelmäßig Fehlermeldungen zu Gesicht. Diese sind bisweilen nervig, aber sie helfen Ihnen, das Problem aufzuspüren, und liefern Informationen zur Fehlerbehebung. Der PHP-Interpreter verfügt über zwei Mechanismen, um mit auftretenden Problemen umzugehen: Fehler (Errors) und Ausnahmen (Exceptions).

- **Fehler** sind Meldungen, die der PHP-Interpreter bei Problemen während der Codeausführung erzeugt. Der PHP-Interpreter hebt hier also quasi die Hand und sagt Ihnen: »Hier stimmt etwas nicht«. Einige Fehler verhindern die Ausführung des Codes auf der Seite, andere nicht.
- **Ausnahmen** sind Objekte, die sowohl vom PHP-Interpreter als auch vom Programmierer erstellt werden können. Diese Objekte werden angelegt, wenn der normalerweise auszuführende Code durch eine Ausnahmesituation an der Ausführung gehindert wird. Wenn ein Exception-Objekt erstellt wird, unterbricht der PHP-Interpreter die Codeausführung und sucht nach einem alternativen Code-Block, der für diese Situation geschrieben wurde; auf diese Weise kann der Code mit dem Problem umgehen und sich davon befreien. Ausnahmen sind so, als würden der PHP-Interpreter oder der Programmierer sagen: »Hier stimmt etwas nicht - gibt es Anweisungen, wie damit umzugehen ist?« Wenn es keinen alternativen Code zur Bewältigung der Situation gibt, wird ein Fehler ausgelöst, der die Ausführung der Seite stoppt.

Sowohl bei Fehlern als auch bei Ausnahmen geben entsprechende Meldungen Aufschluss über Art und Quelle des aufgetretenen Problems. Diese Meldungen können auf der Webseite angezeigt oder in einer Textdatei auf dem Server, der sogenannten Log-Datei, gespeichert werden.

Zusätzlich zu den vom PHP-Interpreter erzeugten Fehlermeldungen kann der Webserver auch eigene Fehlermeldungen erzeugen und an den Browser senden. Dies geschieht, wenn er die vom Browser angeforderte Datei nicht finden kann oder wenn ein anderes Problem den Server lahmlegt.



DIE DARSTELLUNGSWEISE VON PHP-FEHLERN BEEINFLUSSEN

Wenn der PHP-Interpreter bei der Codeausführung auf ein Problem stößt, erzeugt er eine entsprechende Fehlermeldung. Diese kann auf der an den Browser gesendeten Webseite angezeigt oder in einer Datei auf dem Server gespeichert werden.

In der Entwicklungsphase einer Website sollten die vom PHP-Interpreter erzeugten Fehlermeldungen auf der Webseite angezeigt werden, die an den Browser zurückgesendet wird. So können die Programmierenden alle Fehler direkt erkennen und sie dann beheben.

Wird eine Website live geschaltet und es treten Fehler auf, die während der Entwicklung nicht erkannt wurden, sollten die entsprechenden Fehlermeldungen nicht auf der Webseite angezeigt werden, da sie:

- für Besucher nur schwer verständlich sind
- Hackern Hinweise über den inneren Aufbau der Website geben können

Stattdessen zeigt die PHP-Seite eine benutzerfreundliche Meldung an, und die Fehlermeldung wird auf dem Server in Textform in einer sogenannten Log-Datei gespeichert. Diese können die Entwickler überprüfen, um festzustellen, ob nach der Inbetriebnahme der Website irgendwelche Fehler aufgetreten sind.

Für den PHP-Interpreter können drei Einstellungen vorgenommen werden:

- ob Fehler auf dem Bildschirm angezeigt werden sollen oder nicht
- ob sie in eine Log-Datei geschrieben werden sollen oder nicht
- welche Fehler angezeigt werden (beim Erlernen von PHP oder bei der Website-Entwicklung sollten Sie sich alle Fehler anzeigen lassen)

Diese Einstellungen lassen sich entweder in der Datei `php.ini` oder in der Datei `.htaccess` vornehmen (siehe die Seiten 196 bis 199).

- Die Datei `php.ini` enthält die Standardeinstellungen für alle Dateien auf dem Webserver. Wenn sie aktualisiert wird, muss der Server neu gestartet werden, damit die Änderungen wirksam werden.
- In jedem beliebigen Verzeichnis auf dem Webserver kann eine `.htaccess`-Datei abgelegt werden. Sie beeinflusst dann alle Dateien in diesem Verzeichnis und sämtlichen Unterverzeichnissen. Bei Änderungen an `.htaccess`-Dateien muss der Server nicht neu gestartet werden.

php.ini

Die folgenden Einstellungen sollten in der Datei `php.ini` vorgenommen werden. Sie weisen den PHP-Interpreter an, alle Fehler zu melden und sie auf dem Bildschirm und in einer Log-Datei auszugeben:

```
display_errors = On  
log_errors      = On  
error_reporting = E_ALL
```

Wenn eine Website live geht, muss `display_errors` auf `Off` gesetzt werden, damit am Bildschirm keine Fehler mehr angezeigt werden.

```
display_errors = Off
```

.htaccess

Die folgenden Einstellungen können in eine `.htaccess`-Datei eingefügt werden, damit der PHP-Interpreter alle Fehler meldet, auf dem Bildschirm darstellt und in eine Log-Datei schreibt:

```
php_flag display_errors On  
php_flag log_errors      On  
php_value error_reporting -1
```

Wenn eine Website live geht, muss `display_errors` auf `Off` gesetzt werden, damit keine Fehlermeldungen mehr im Browserfenster erscheinen.

```
php_flag display_errors Off
```

Im Code-Download zu diesem Buch werden die Einstellungen des PHP-Interpreters über .htaccess-Dateien gesteuert.

Einige Ordner haben ihre eigenen .htaccess-Dateien, um die Einstellungen für die jeweiligen Beispiele zu steuern.

Die Code-Beispiele in diesem Kapitel befinden sich in zwei Ordner: einer für in der Entwicklung befindliche Websites und einer für aktive Websites.

PHP

section_b/c10/development/.htaccess

```
① [php_flag display_errors On  
     php_flag log_errors On  
     php_value error_reporting -1]
```

PHP

section_b/c10/live/.htaccess

```
② [php_flag display_errors Off  
     php_flag log_errors - On  
     php_value error_reporting -1]
```

PHP

section_b/c10/development/find-error-log.php

Your error log is stored here:
③ <?= ini_get('error_log') ?>

PHP

section_b/c10/development/sample-error.php

```
<?php  
④ echo 'Finding an error';  
?>
```

ERGEBNIS

Parse error: syntax error, unexpected string content "Finding an error";" in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/sample-error.php on line 2

ERGEBNIS

```
[27-Jan-2021 14:41:13 UTC] PHP Parse error:  syntax error,  
unexpected string content "Finding an error";" in  
/Users/Jon/Sites/localhost/phpbook/section_b/c10/  
development/sample-error.php on line 2
```

Hinweis: Log-Dateien sollten oberhalb des Dokumentenstammverzeichnisses gespeichert werden (siehe Seite 526), um zu verhindern, dass Hacker deren Pfad erraten und sie über eine URL abrufen.

1. Diese Einstellungen sollten verwendet werden, solange sich eine Website in der Entwicklung befindet. Sie weisen den PHP-Interpreter an, alle Fehler auf dem Bildschirm anzuzeigen und sie außerdem in eine Log-Datei zu schreiben.

2. Wenn die Website live geht, sollten keine Fehlermeldungen mehr im Browser angezeigt werden. Stattdessen werden sie in eine Log-Datei geschrieben, damit das Entwicklungsteam erkennen kann, ob bei Nutzern Fehler auftreten sind.

3. Webserver können Log-Dateien in verschiedenen Ordner ablegen. Um herauszufinden, wo auf Ihrem Server sich die Log-Datei für Fehler befindet, verwenden Sie die PHP-Funktion ini_get().

4. Diese einfache PHP-Seite erzeugt einen Fehler, weil die Anführungszeichen nicht zusammenpassen.

Die bei der Ausführung dieser Datei vom PHP-Interpreter erzeugte Fehlermeldung sehen Sie nebenan.

Der erste Ergebniskasten zeigt die Fehlermeldung im Browser für den obigen Fehler. Wie Sie sehen können, ist die Meldung nicht allzu benutzerfreundlich. Die nächsten acht Seiten helfen Ihnen zu verstehen, was diese Fehlermeldungen bedeuten.

Der zweite Ergebniskasten zeigt den Inhalt der Fehlerprotokolldatei an. Sie können die Log-Datei mit einem Text- oder Code-Editor öffnen. Die Fehlermeldung entspricht der auf dem Bildschirm angezeigten, jedoch werden ihr das Datum und die Uhrzeit vorangestellt, zu der der Fehler gemeldet wurde.

FEHLERQUELLEN RICHTIG DEUTEN

Die vom PHP-Interpreter erzeugten Fehlermeldungen können auf den ersten Blick verwirrend aussehen, aber sie haben alle die gleiche Struktur und enthalten vier Informationen, die Entwicklern beim Aufspüren der Fehlerursache helfen.

Programmierer sprechen davon, dass der PHP-Interpreter Fehler »wirft«, wenn er sie findet. Die Fehlermeldungen haben die unten gezeigte Struktur und bestehen aus vier Datenelementen:

- Die ersten beiden (Level und Beschreibung) beschreiben den aufgetretenen Fehler.
- Die nächsten beiden (der Dateipfad und die Zeilennummer) sagen Ihnen, wo Sie mit der Suche nach dem Problem beginnen müssen.

Die Fehlerstufe (oder das Error Level) beschreibt die allgemeine Art des vom PHP-Interpreter festgestellten Problems.

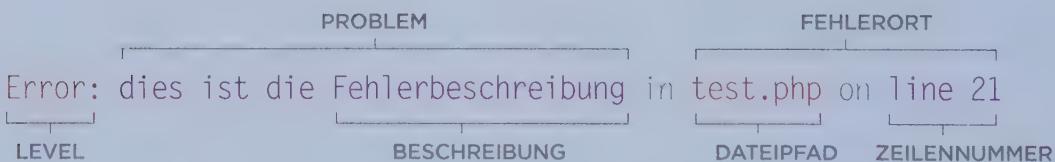
Die Beschreibung erläutert den Fehler genauer.

Der Dateipfad ist der Pfad zur Datei, in der der Fehler gefunden wurde.

Die Zeilennummer ist die Zeile in dieser Datei, in der der Fehler gefunden wurde.

Die wichtigsten Fehlerarten, die Sie wahrscheinlich antreffen werden, sind auf der rechten Seite beschrieben. Auf den nächsten sechs Seiten finden Sie PHP-Dateien mit Beispielen für jeden dieser Fehlerarten sowie Beschreibungen, die Ihnen zeigen, wie Sie den jeweiligen Fehler aufspüren und beheben können.

Einige Fehler werden bereits früher verursacht, als der PHP-Interpreter den Fehler meldet, aber der Dateiname und die Zeilennummer verraten Ihnen, wo Sie mit der Suche beginnen müssen.



FEHLER-LEVEL UND -ARTEN

Die wichtigsten Arten von PHP-Fehlern sind unten aufgeführt. Einige Fehler unterbrechen die Code-Ausführung durch den PHP-Interpreter und müssen behoben werden, damit die Seite funktioniert. Andere erscheinen wiederum eher wie Verbesserungsvorschläge, sollten aber trotzdem behoben werden.

PARSE

Seiten 356 und 357

Ein Parse-Fehler bedeutet, dass die Syntax des PHP-Codes fehlerhaft ist. Somit kann der PHP-Interpreter die Datei nicht korrekt lesen oder parsen, sodass er nicht versuchen wird, den Code auszuführen.

Wenn die Einstellungen des PHP-Interpreters die Anzeige von Fehlern auf dem Bildschirm vorsehen, wird ausschließlich dieser Fehler angezeigt. Werden keine Fehler auf dem Bildschirm angezeigt, sitzt der Besucher vor einer leeren Seite.

Parse-Fehler müssen behoben werden, damit die Seite ausgeführt und etwas anderes als eine Fehlermeldung angezeigt werden kann.

Hinweis: Fehler-Level und -meldungen verändern sich über verschiedene PHP-Versionen hinweg. In diesem Kapitel sehen Sie die Fehlermeldungen von PHP 8.

FATAL

Seiten 358 und 359

Ein fatales Fehler zeigt an, dass der PHP-Interpreter die Syntax des PHP-Codes zunächst für gülig befand. Er hat also versucht, den Code auszuführen, aber irgend etwas hat die korrekte Ausführung dann verhindert.

Der PHP-Interpreter stoppt in der Code-Zeile, in der er einen schwerwiegenden Fehler findet. Das heißt, dass Benutzer möglicherweise einen Teil einer Seite sehen, der bereits erstellt wurde, bevor der PHP-Interpreter den Fehler entdeckt hat.

Seit PHP 7 wird bei den meisten fatalen Fehlern ein Exception-Objekt erzeugt. Dies gibt dem Programmierer die Möglichkeit, den Fehler zu beheben.

Wenn Sie einen Fehler sehen, der mit der Fehlerstufe Deprecated (veraltet) ausgezeichnet ist, bedeutet dies, dass eine Funktion in absehbarer Zukunft aus PHP entfernt werden soll.

NON-FATAL

Seiten 360 und 361

Bei einem solchen nicht schwerwiegenden Fehler wird eine Fehlermeldung erzeugt, die auf ein mögliches Problem hinweist, der Code wird jedoch weiter ausgeführt.

- Ein Warning ist ein nicht schwerwiegender Fehler, der darauf hinweist, dass der PHP-Interpreter einen Fehler gefunden hat, der möglicherweise zu einem Problem führt.

- Ein Notice ist ein nicht schwerwiegender Fehler, der auf mögliche Probleme hinweist, die der PHP-Interpreter gefunden hat.

In PHP 8 wurden viele Notices in Warnings umgewandelt.

Fehler mit dem Level Strict enthalten Ratschläge, wie Sie Ihren PHP-Code besser schreiben können.

PARSE-FEHLER

Ein Parse-Fehler entsteht durch ein Problem mit der Code-Syntax. Er verhindert die Darstellung einer Seite, weil der PHP-Interpreter den Code nicht versteht. Parse-Fehler müssen Sie beheben, um die Code-Ausführung zu ermöglichen.

Parse-Fehler entstehen häufig durch Tippfehler, etwa durch ein falsches Anführungszeichen oder ein fehlendes Semikolon oder eine fehlende Klammer. Diese einfachen Fehler können bewirken, dass der PHP-Interpreter den Code nicht lesen kann.

In Zeile 2 dieses Beispiels stimmen die Anführungszeichen der Variablen nicht überein (es handelt sich um ein einfaches und ein doppeltes Anführungszeichen).

Laut Fehlermeldung tritt das Problem in Zeile 3 auf. Das liegt daran, dass der PHP-Interpreter den Fehler erst beim Auftreten eines weiteren einfachen Anführungszeichens bemerkt hat.

Als er in Zeile 3 auf ein weiteres einfaches Anführungszeichen stieß, behandelte er dieses so, als würde es die Variable aus Zeile 2 schließen.

Laut Fehlermeldung gibt es in Zeile 3 den unerwarteten Bezeichner 'pencil', denn nach dem zweiten einfachen Anführungszeichen steht das Wort pencil.

Um Parse-Fehler zu beheben, gehen Sie in die Zeile, in der der Fehler aufgetreten ist. Lesen Sie die Zeile von links nach rechts und überprüfen Sie jede darin enthaltene Anweisung. Wenn Sie das Problem nicht erkennen können, sehen Sie sich die vorhergehende Zeile an.

Wenn `display_errors` auf Off gesetzt ist, führt ein Parse-Fehler zur Anzeige eines leeren Bildschirms. Parse-Fehlern müssen Sie finden und beheben, bevor die Seite etwas anzeigen kann.

section_b/c10/development/parse-error-1.php

PHP

```
1 <?php
2 $username = 'Ivy';
3 $order    = ['pencil', 'pen', 'notebook',];
4 ?>
5 <h1>Basket</h1>
6 <?= $username ?>
7 <?php foreach ($order as $item) { ?>
8     <?= $item ?><br>
9 <?php } ?>
```

ERGEBNIS

Parse error: syntax error, unexpected identifier "pencil" in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-1.php on line 3

Hinweis: Wenn Sie einen Code-Editor mit Syntaxmarkierung verwenden, liefern Ihnen die Farben im Code oft einen Hinweis auf die Position eines Syntaxfehlers.

PHP

section_b/c10/development/parse-error-2.php

```

1 <?php
2 $username = 'Ivy'
3 $order    = ['pencil', 'pen', 'notebook',];
4 ?> ...

```

ERGEBNIS

Parse error: syntax error, unexpected variable "\$order" in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-2.php on line 3

PHP

section_b/c10/development/parse-error-3.php

```

1 <?php
2 $username = 'Ivy';
3 $order    = ['pencil', 'pen', 'notebook',];
4 ?> ...

```

ERGEBNIS

Parse error: Unclosed '[' does not match ')' in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-3.php on line 3

PHP

section_b/c10/development/parse-error-4.php

```

1 <?php
2 $username = 'Ivy';
3 order    = ['pencil', 'pen', 'notebook',];
4 ?> ...

```

ERGEBNIS

Parse error: syntax error, unexpected identifier "order" in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-4.php on line 3

Um einen Parse-Fehler zu finden, kommentieren Sie die zweite Hälfte der Seite aus. Wenn Sie immer noch denselben Fehler sehen, liegt das Problem in der ersten Seitenhälfte. Wenn nicht, liegt es in der zweiten Hälfte. Wiederholen Sie den Vorgang, um die Ursache weiter einzuschränken.

Am Ende von Zeile 2 sollte ein Semikolon stehen. Die Fehlermeldung beklagt die unerwartete Variable \$order in Zeile 3, weil die vorherige Anweisung nicht mit einem Semikolon abgeschlossen wurde.

Wie die ersten beiden Beispiele zeigen, liegt die Ursache für einen Parse-Fehler oft in einer Zeile, die vor der Zeile liegt, in der der Fehler gefunden wurde. Wenn Sie das Problem in der bemängelten Zeile nicht erkennen können, sehen Sie sich die vorherige Code-Zeile an, die ausgeführt wurde.

In Zeile 3 wird ein Array erstellt. Es beginnt mit einer eckigen öffnenden Klammer, endet aber mit einer schließenden runden Klammer.

Hier verweist die Fehlermeldung auf die richtige Zeile und nennt das genaue Problem: Nicht geschlossenes '[' passt nicht zu ')' in Zeile 3.

Der Variablenname in Zeile 3 beginnt nicht mit einem \$-Symbol.

Die Fehlermeldung besagt, dass der Bezeichner 'order' in Zeile 3 unerwartet ist, weil order weder ein Schlüsselwort noch eine Anweisung ist, die dem PHP-Interpreter bekannt wären (er weiß nicht, dass es sich um einen Variablennamen handeln soll, weil kein \$-Symbol davor steht).

Probieren Sie es: Um herauszufinden, ob Sie die Probleme in diesem Abschnitt verstanden haben, versuchen Sie, die in der Datei beschriebenen Fehler zu beheben, und führen die Beispiele dann erneut aus.

SCHWERWIEGENDE FEHLER

Ein schwerwiegender **Fatal Error** wird ausgelöst, wenn der PHP-Interpreter ein Problem feststellt, das die weitere Ausführung des Codes verhindert. Das bedeutet, dass die Benutzer nur einen Teil der Seite sehen, und zwar bis zu dem Punkt, an dem der schwerwiegende Fehler gefunden wurde.

Wenn Sie einen schwerwiegenden Fehler sehen, hielt der PHP-Interpreter die Syntax zunächst für korrekt, fand dann aber ein Problem, das ihn an der weiteren Code-Ausführung hinderte.

In Zeile 4 versucht dieses Beispiel, eine Ganzzahl (die in Zeile 2 in \$price gespeichert wurde) mit einer Zeichenkette (die in Zeile 3 in \$quantity gespeichert wurde) zu multiplizieren.

Die Fehlermeldung lautet »Unsupported operand types int * string« und weist darauf hin, dass eine ganze Zahl nicht mit einer Zeichenkette multipliziert werden kann. Das Problem wird noch vor der Anzeige irgendwelcher Inhalte entdeckt, sodass der Besucher die Überschrift Basket und den Text Total: nicht zu sehen bekommt.

Um zu verhindern, dass dieser Fehler erneut auftritt, könnte die Seite die beiden Werte vor dem Multiplikationsversuch zunächst validieren.

Bis PHP 7.4 hätte dieses Beispiel statt eines schwerwiegenden Fehlers eine Warnung erzeugt.

Wenn die HTML-Seite vor dem Auftreten des Fehlers bereits teilweise erstellt wurde, kann der Benutzer möglicherweise den vor dem Fehler befindlichen Teil der Seite sehen. Ist dies nicht der Fall, wird eventuell auch eine leere Seite angezeigt.

Bei schwerwiegenden Fehlern müssen Sie herausfinden, warum der PHP-Interpreter den Code nicht abarbeiten kann, und das Problem beheben, um die Anzeige der gesamten Seite zu ermöglichen.

section_b/c10/development/fatal-error-1.php

PHP

```
1 <?php  
2 $price      = 7;  
3 $quantity   = 'five';  
4 $total      = $price * $quantity;  
5 ?>  
6 <h1>Basket</h1>  
7 Total: $<?= $total ?>
```

ERGEBNIS

Fatal error: Uncaught TypeError: Unsupported operand types: int * string in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-1.php:4 Stack trace: #0 {main} thrown in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-1.php on line 4

Hinweis: Vor PHP 7 konnte sich eine Seite nicht von einem schwerwiegenden Fehler erholen. Seit PHP 7 wird bei fatalen Fehlern ein Exception-Objekt erzeugt, über das sich der Code mit dem Problem auseinandersetzen kann (siehe die Seiten 372 und 373). Werden Exceptions nicht behandelt (oder abgefangen), dann werden daraus Fatal Errors, weshalb diese Fehlermeldungen mit den Worten Uncaught error beginnen.

PHP

```
section_b/c10/development/fatal-error-2.php

1 <?php
2 function total(int $price, int $quantity) {...}
3 ?
4 <h2>Basket</h2>
5 <?= totals(3, 5) ?>
```

ERGEBNIS**Basket**

Fatal error: Uncaught Error: Call to undefined function totals() in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-2.php:8 Stack trace: #0 {main} thrown in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-2.php on line 8

PHP

```
section_b/c10/development/fatal-error-3.php
```

```
1 <?php
2 function total(int $price, int $quantity) {...}
3 ?
4 <h2>Basket</h2>
5 <?= total(3) ?>
```

ERGEBNIS**Basket**

Fatal error: Uncaught ArgumentCountError: Too few arguments to function total(), 1 passed in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-3.php on line 8 and exactly 2 expected in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-3.php:2 Stack trace: #0 /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-3.php(8): total(3) #1 {main} thrown in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/fatal-error-3.php on line 2

PHP

```
section_b/c10/development/fatal-error-4.php
```

```
1 <?php $basket = new Basket(); ?><h2>Basket</h2>
```

ERGEBNIS

Fatal error: Uncaught Error: Class 'Basket' not found in /Applications/MAMP/htdocs/phpbook/section_b/c10/development/fatal-error-4.php:1 Stack trace: #0 {main} thrown in /Applications/MAMP/htdocs/phpbook/section_b/c10/development/fatal-error-4.php on line 1

In Zeile 2 wird die Funktion total() deklariert (die vollständige Funktionsdefinition finden Sie im Code-Download). In Zeile 8 ruft der Code die Funktion totals() auf.

Die Fehlermeldung lautet Call to undefined function totals(), weil die Funktion totals() nicht existiert; eigentlich sollte total() aufgerufen werden. Die Überschrift Basket wird angezeigt, weil der PHP-Interpreter die Code-Ausführung erst nach Feststellung des Fehlers abbricht.

In Zeile 8 wird die Funktion total() aufgerufen, aber mit nur einem Argument (anstatt mit zweien).

Die Fehlermeldung besagt: Zu wenige Argumente für Funktion total(), weil die Funktion zwei Argumente benötigt.

Sie besagt auch, dass ein Argument übergeben wurde, obwohl genau zwei erwartet werden. Um dies zu beheben, muss die Funktion mit der richtigen Anzahl von Argumenten aufgerufen werden.

In Zeile 1 wird ein Objekt mit einer Klasse Basket erstellt, aber die Klassendefinition wurde nicht in die Seite eingebunden. Die Meldung besagt: Klasse 'Basket' nicht gefunden. Der Rest der Seite kann nicht mehr angezeigt werden, weil das Objekt nicht erstellt werden kann. Um dieses Problem zu beheben, muss zuerst die Klassendefinition eingebunden werden.

Hinweis: Wenn innerhalb einer Funktion oder Methode ein Ausnahmeobjekt erstellt (oder geworfen) wird, gibt das in der Fehlermeldung angezeigte Stack-Trace den Namen der Datei und die Code-Zeile an, die diese Funktion oder Methode aufgerufen hat.

NICHT SCHWERWIEGENDE FEHLER (WARNING ODER NOTICE)

Nicht schwerwiegende Fehler werden ausgelöst, wenn der PHP-Interpreter glaubt, dass es ein Problem geben könnte, er aber dennoch versucht, den verbleibenden Code weiter auszuführen.

Eine Warnung (Warning) zeigt an, dass ein Fehler vorliegt, der wahrscheinlich ein Problem verursacht, und ein Hinweis (Notice) signalisiert, dass ein Fehler vorliegen könnte.

In diesem Beispiel werden drei Variablen deklariert:

- In Zeile 2 wird \$price deklariert. Ihr Wert ist die Zahl 7.
- In Zeile 3 wird \$quantity deklariert. Ihr Wert ist eine Zeichenkette mit dem Inhalt '0a'.
- In Zeile 4 wird \$total deklariert. Als Wert sollte ihr der Wert in \$price multipliziert mit dem Wert in \$quantity zugewiesen werden.

Zeile 4 erzeugt die Warnmeldung, dass ein nicht-numerischer Wert gefunden wurde, weil der in \$quantity gespeicherte Wert ein String ist.

Da das erste Zeichen des Strings die Zahl 0 ist, versucht der PHP-Interpreter, das erste Zeichen 0 zu verwenden (und ignoriert den Rest des Strings – siehe Seite 361). Dann läuft die Seite weiter und zeigt den in \$total gespeicherten Wert an.

In beiden Fällen handelt es sich um keine schwerwiegenden (fatalen) Fehler. Sie werden ausgelöst, wenn der PHP-Interpreter sie erkennt, aber sie halten die Ausführung der Seite nicht an.

Alle Fehler sollten korrigiert werden, da sie gravierende Auswirkungen auf den Rest der Seite haben können (wie das folgende Beispiel zeigt).

section_b/c10/development/warning-1.php

PHP

```
1 <?php  
2 $price    = 7;  
3 $quantity = '0a';  
4 $total    = $price * $quantity;  
5 ?>  
6 <h1>Basket</h1>  
7 Total: $<?= $total ?>
```

ERGEBNIS

Warning: A non-numeric value encountered in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/warning-1.php on line 4

Basket

Total: \$0

Das könnte für die Website zu einem ernsthaften Problem werden, weil der Gesamtbetrag jetzt mit null Dollar ausgewiesen wird.

Sie können die PHP-Funktion var_dump() (Seite 193) verwenden, um Variablenwerte und deren Datentyp zu überprüfen.

PHP

section_b/c10/development/warning-2.php

```

1 <?php $list = false; ?>
2 <h1>Basket</h1>
3 <?php foreach ($list as $item) { ?>
4     Item: <?= $item ?><br>
5 <?php } ?>
```

ERGEBNIS

Basket

Warning: foreach() argument must be of type array|object, bool given in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/warning-2.php on line **3**

PHP

section_b/c10/development/warning-3.php

```

1 <?php include 'header.php'; ?>
2 <h1>Basket</h1>
```

ERGEBNIS

Warning: include(header.php): Failed to open stream: No such file or directory in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/warning-3.php on line **1**

Warning: include(): Failed opening 'header.php' for inclusion
 (include_path='..:/Applications/MAMP/bin/php/php8.0.0/lib/php') in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/warning-3.php on line **1**

Basket

PHP

section_b/c10/development/warning-4.php

```

1 <?php $list = ['pencil', 'pen', 'notebook',]; ?>
2 <?= $list ?>
```

ERGEBNIS

Warning: Array to string conversion in
/Users/Jon/Sites/localhost/phpbook/section_b/c10/development/warning-4.php on line **2**
 Array

Die \$list-Variable in Zeile 1 sollte ein Array enthalten, aber ihr wurde der boolesche Wert false zugewiesen. In Zeile 3 durchläuft eine foreach-Schleife dann das Array der Artikel in \$list.

Die Fehlermeldung besagt, dass das Argument für foreach() vom Typ array|object sein muss, weil die Schleife nicht über einen booleschen Wert hinweglaufen kann.

Hinweis: Wenn foreach mit einem leeren Array oder mit einem eigenschaftslosen Objekt verwendet wird, entsteht dabei kein Fehler.

In Zeile 1 wird ein Header eingebunden, aber die Include-Datei ist nicht auffindbar. Infolgedessen werden zwei Fehlermeldungen angezeigt:

- Stream konnte nicht geöffnet werden: Die Datei oder das Verzeichnis konnten nicht gefunden werden.
- Failed opening ... for inclusion bedeutet, dass die Datei nicht eingebunden werden konnte.

Der PHP-Interpreter versucht nach einer fehlenden Include-Datei, den restlichen Code anzuzeigen.

In Zeile 1 wird der Variablen \$list ein Array zugewiesen. In Zeile 2 dient die Kurzform des echo-Befehls dazu, den Inhalt der Variablen \$list auszugeben.

Dies erzeugt die Fehlermeldung **Array to string conversion**. Sie zeigt an, dass die Seite versucht, ein Array in einen String zu konvertieren, was aber nicht gelingt, sodass der Inhalt nicht angezeigt werden kann. (Bis PHP 8 war dies ein Hinweis und keine Warnung.)

DEBUGGING: AUF FEHLERSUCHE GEHEN

Eine Website sollte gründlich getestet und sämtliche Fehler sollten korrigiert werden, bevor sie online geht. Wenn der Fehler nicht in der in der Meldung angegebenen Zeile liegt, können Sie ihn mit verschiedenen Techniken aufspüren.

1. Die Ausgabe von Hinweisen auf dem Bildschirm zeigt, wie weit der Interpreter gekommen ist, bevor ein Fehler auftritt.

Der echo-Befehl dient zur Ausgabe von Hinweisen in den Zeilen 2, 9, 17, 24.

Die Meldung in Zeile 9 wird nur angezeigt, wenn die Funktion total() aufgerufen wird (so wie in Zeile 18).

2. Durch das Auskommentieren einzelner Codeabschnitte lässt sich der potenziell problembehaftete Code eingrenzen. In den Zeilen 20 und 23 werden die Kopf- und Fußzeile auskommentiert, um nachzuweisen, dass der Fehler nicht in diesen Dateien liegt. Sie können auch Funktionsaufrufe auskommentieren (und ihre Rückgabewerte fest einprogrammieren), um zu prüfen, ob die Fehler in einer Funktion auftreten.

3. Die PHP-Funktion var_dump() gibt die in Variablen gespeicherten Werte und ihren Datentyp aus, sodass Sie überprüfen können, ob eine Variable den von Ihnen erwarteten Wert enthält. Auf diese Weise wird in Zeile 26 der Wert von \$basket überprüft. Dabei zeigt sich, dass der Wert des dritten Elements im Array \$basket ein String und keine Zahl ist.

ERGEBNIS

1: Start of page

2: Before function called

3: Inside total() function

Warning: A non-numeric value encountered in /Applications/MAMP/htdocs/phpbook/section_b/c10/development/tracking-down-errors.php on line 12

Basket

Total: \$2.00

4: End of page

```
$basket: array(3) { ["pen"]=> float(1.2) ["pencil"]=> float(0.8) ["paper"]=> string(3) "two" }
```

Test total() function:

3: Inside total() function

```

1 <?php
① 2 echo '<p><i>1: Start of page</i></p>';
3 $basket['pen']      = 1.20;
4 $basket['pencil']   = 0.80;
5 $basket['paper']    = 'two';
6
7 function total(array $basket): int
8 {
① 9 echo '<p><i>3: Inside total() function</i></p>';
10    $total = 0;
11    foreach ($basket as $item => $price) {
12        $total = $total + $price;
13    }
14    return $total;
15 }
16
① 17 echo '<p><i>2: Before function called</i></p>';
18 $total = total($basket);
19 ?>
② 20 <?php // include 'header.php' ?>
21 <h3>Basket</h3>
22 <p><b>Total: $<?= number_format($total, 2) ?></b></p>
② 23 <?php // include 'footer.php' ?>
① 24 <?php echo '<p><i>4: End of page</i></p>'; ?>
25 <hr><!-- All remaining code is test code -->
③ 26 <p><b>$basket:</b> <?= var_dump($basket) ?></p>
27 <b>Test total() function:</b>
28 <?php
29 $testbasket['pen']      = 1.20;
④ 30 $testbasket['pencil']   = 0.80;
31 $testbasket['paper']    = 2;
32 ?>
33 <?= total($testbasket) ?>

```

4. Sie können auch Testszenarien für Funktionen und Methoden schreiben, um zu prüfen, ob sie die erwartete Aufgabe erfüllen.

Wenn ein Skript Funktionen oder Methoden benutzt, ist es sinnvoll, jede einzelne isoliert zu testen (anstatt im Kontext der gesamten Seite).

Sie können sehen, dass am Ende der Seite ein Test für die Funktion `total()` erfolgt. Das Array `$testbasket` wird mit Werten zum Testen der Funktion erstellt. Anschließend wird die Funktion `total()` aufgerufen, um zu prüfen, ob sie den richtigen Wert zurückgibt.

Beachten Sie, dass die `echo`-Meldung in Zeile 9 erneut angezeigt wird, wenn die Funktion `total()` in Zeile 33 ein zweites Mal aufgerufen wird.

Manche Programmierer schreiben einfache Seiten zum Testen jeder einzelnen Funktion (anstatt den Code im Falle eines Problems zu debuggen). Wenn Sie wissen, dass eine Funktion für sich genommen funktioniert, sollten die Anweisungen in der Funktion nicht die Ursache des Problems sein. Somit können Sie sich auf die Werte konzentrieren, die an die Funktion übergeben werden.

Wenn die an die Funktion übergebenen Werte nicht korrekt zu sein scheinen, können Sie zurückverfolgen, woher diese stammen, und so den Ursprung des Fehlers finden.

Hinweis: Die Einrückung von Code in geschweiften Klammern um vier Leerzeichen erleichtert das Auffinden von Problemen wie einer fehlenden schließenden Klammer.

Im Schritt 1 mit dem Echo-Befehl ausgegebene Werte werden nicht eingerrückt. So lässt sich leichter erkennen, wo sie hinzugefügt wurden.

Einige PHP-Editoren können jede Code-Zeile einzeln durchlaufen; dies wird als »schrittweise« Code-Verarbeitung bezeichnet. In jeder Zeile können Sie die Werte überprüfen, um herauszufinden, wo eventuell ein Fehler im Code vorliegt. Sie können auch Haltepunkte setzen, an denen der Code nicht weiterlaufen soll (und an dieser Stelle dann den Inhalt von Variablen überprüfen).

EINE WEBSITE LIVE SCHALTEN

Wenn eine Website bereit zur Veröffentlichung ist, sollten die Einstellungen dahingehend geändert werden, dass der PHP-Interpreter keine Fehlermeldungen mehr auf den Webseiten anzeigt. Speichern Sie stattdessen die Fehlermeldungen in einer Log-Datei, die Sie dann in regelmäßigen Abständen überprüfen.

Selbst bei sorgfältig getesteten Websites können Fehler übersehen worden sein, oder es treten Probleme mit dem Webhosting auf. Nutzen Sie daher auf dem Live-Server die Einstellungen in den Dateien `php.ini` oder `.htaccess`, um:

- die Anzeige von Fehlermeldungen auf dem Bildschirm zu verhindern und
- Fehlermeldungen in einer Log-Datei zu speichern.

Log-Dateien können mit einem Text- oder Code-Editor geöffnet werden. Die Meldungen in der Log-Datei beginnen mit dem Datum und der Uhrzeit des Fehlerereignisses, und es folgen Meldungen, wie sie ansonsten auch auf dem Bildschirm angezeigt werden. Wenn Sie die bisherigen Beispiele in diesem Kapitel ausgeführt haben, enthält die Log-Datei die unten gezeigten Fehlermeldungen.

Fehler sollten auf einer Entwicklungsversion der Webseite (auf einem Testserver oder Ihrem lokalen Rechner) korrigiert werden, nicht auf der aktiven Website. Für jeden Fehler in der Log-Datei:

1. versuchen Sie, den Fehler zu reproduzieren, damit Sie wissen, was die Ursache für aufgezeichnete Meldung war.
2. verwenden Sie die auf den vorhergehenden Seiten gezeigten Techniken, um den fehlerverursachenden Code zu finden.
3. korrigieren Sie den fehlerhaften Code.

Sobald das Problem behoben wurde, sollte die Website nochmals getestet werden, bevor Sie die neue Version auf die Live-Website hochladen, da durch die Fehlerbehebung möglicherweise neue Probleme entstanden sind.

```
[27-Jan-2021 14:56:44 UTC] PHP Parse error: syntax error, unexpected string content "Finding an error%;" in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/sample-error.php on line 2
[27-Jan-2021 14:56:51 UTC] PHP Parse error: syntax error, unexpected identifier "pencil" in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-1.php on line 3
[27-Jan-2021 14:57:02 UTC] PHP Parse error: syntax error, unexpected variable "$order" in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-2.php on line 3
[27-Jan-2021 14:57:04 UTC] PHP Parse error: Unclosed '[' does not match ']' in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-3.php on line 3
[27-Jan-2021 14:57:06 UTC] PHP Parse error: syntax error, unexpected identifier "order" in /Users/Jon/Sites/localhost/phpbook/section_b/c10/development/parse-error-4.php on line 3
```

Der PHP-Interpreter kann Fehler auch in einer Datenbank speichern, aber das wird in diesem Buch nicht behandelt, da Anfänger (und die meisten Websites) sie in einer Log-Datei speichern.

Log-Dateien können viel Festplattenspeicher auf dem Webserver belegen, weshalb der Server-Administrator sie regelmäßig archivieren oder löschen muss.

FUNKTIONEN ZUR FEHLERBEHANDLUNG

Wenn der PHP-Interpreter einen schwerwiegenden oder nicht schwerwiegenden Fehler auslöst, kann er eine benutzerdefinierte Funktion, den sogenannten Error-Handler, aufrufen. Auf aktiven Websites kann diese Funktion verhindern, dass Benutzer eine leere oder abrupt abgebrochene Seite sehen.

NICHT SCHWERWIEGENDE FEHLER BEHANDELN

Wenn der PHP-Interpreter einen nicht schwerwiegenden Fehler meldet, führt der den Rest der Datei weiter aus. Das kann zu ernsthaften Problemen führen. Auf Seite 360 zum Beispiel hat ein Fehler zu einem Bestellwert von 0 Dollar geführt.

Zur Fehlerbehebung muss normalerweise der Code geändert werden. Solange sie nicht behoben wurden, können Sie mit einer Fehlerbehandlungsfunktion versuchen, mit nicht schwerwiegenden Fehlern umzugehen.

Die PHP-Funktion `set_error_handler()` teilt dem PHP-Interpreter den Namen der benutzerdefinierten Funktion mit, die beim Auftreten eines nicht fatalen Fehlers aufgerufen werden soll. Beim Übergeben des entsprechenden Funktionsnamens an `set_error_handler()` werden die nachfolgenden Klammern weggelassen.

Diese Funktion könnte zum Beispiel eine benutzerfreundliche Meldung anzeigen und dann die weitere Code-Ausführung stoppen.

In einem zweiten Parameter können Sie angeben, für welche Fehler-Level der Error-Handler zuständig ist, aber beim Erlernen von PHP ist es besser, ihn für alle nicht schwerwiegenden Fehler zu verwenden.

`set_error_handler('name')`

FUNKTION ZUR
BEHANDLUNG NICHT
SCHWERWIEGENDER
FEHLER

SCHWERWIEGENDE FEHLER BEHANDELN

Schwerwiegende Fehler halten die Seite an, und die in `set_error_handler()` angegebene Funktion wird nicht mehr ausgeführt.

Seit PHP 7 hat der PHP-Interpreter die meisten schwerwiegenden Fehler in Exceptions umgewandelt. Mehr über Ausnahmen und deren Behandlung erfahren Sie nach dem nächsten Beispiel. Wird ein schwerwiegender Fehler jedoch in eine Ausnahme umgewandelt und bleibt danach unbehandelt, wird er wieder zu einem schwerwiegenden Fehler.

Es ist möglich, eine benutzerdefinierte sogenannte Shutdown-Funktion zu benennen, die immer dann aufgerufen wird, wenn eine Seite fertig abgearbeitet wurde, der Exit-Befehl zum Einsatz kam oder ein schwerwiegender Fehler die Ausführung der Seite beendet.

Die Shutdown-Funktion kann prüfen, ob bei der Ausführung der Seite ein Fehler aufgetreten ist, und, falls dies der Fall ist, eine benutzerfreundliche Fehlermeldung anzeigen und den Fehler protokollieren. Die PHP-Funktion `register_shutdown_function()` teilt dem PHP-Interpreter den Namen der Funktion mit, die aufgerufen werden soll, wenn eine Seite aufhört zu laufen (sie wird auf den Seiten 376 und 377 verwendet). Beachten Sie, dass bei der Angabe des Namens der Shutdown-Funktion keine nachfolgenden Klammern verwendet werden sollten.

`register_shutdown_function('name')`

FUNKTION ZUR
BEHANDLUNG
SCHWERWIEGENDER
FEHLER

FUNKTION ZUR BEHANDLUNG NICHT SCHWERWIEGENDER FEHLER

Unentdeckte nicht schwerwiegende Fehler können zu Problemen auf einer Online-Website führen. Wenn ein Fehler auftritt, protokolliert diese Fehlerbehandlungsfunktion den Fehler, zeigt eine benutzerfreundliche Fehlermeldung an und bricht die Code-Ausführung ab.

In diesem Beispiel legt die PHP-Funktion `set_error_handler()` fest, dass der PHP-Interpreter beim Auftreten eines nicht schwerwiegenden Fehlers die Funktion `handle_error()` aufrufen soll. Verwendet eine Website eine eigene Funktion zur Behandlung nicht schwerwiegender Fehler, dann wendet der PHP-Interpreter seinen eigenen Fehlerbehandlungscode nur dann an, wenn diese Funktion den Wert `false` zurückgibt. Und wenn der Fehlerbehandlungscode des PHP-Interpreters nicht ausgeführt wird, wird der Fehler auch nicht in der Log-Datei protokolliert (sodass die Entwickler nicht wissen, dass er aufgetreten ist).

Diese Fehlerbehandlungsfunktion darf nicht den Wert `false` zurückgeben, da der Rest der Seite sonst nicht weiterläuft (nachdem sie eine benutzerfreundliche Fehlermeldung angezeigt hat).

Die Funktion muss die Fehlermeldung also selbst in der Log-Datei speichern, bevor die Seite aufhört zu laufen. Wenn der PHP-Interpreter eine Funktion zur Fehlerbehandlung aufruft, übergibt er ihr vier Argumente mit Informationen über den Fehler (den Werten, die sonst in der Fehlermeldung erscheinen würden):

- Die Fehlerstufe (als Ganzzahl)
- Eine Fehlermeldung (als String)
- Den Pfad zur Datei, in der der Fehler aufgetreten ist
- Die Code-Zeile, in der der Fehler entdeckt wurde

In der Funktionsdefinition von `handle_error()` müssen entsprechende Parameter bereitgestellt werden, um diese Werte verwenden zu können.

In diesem Beispiel werden die Daten über den Fehler zur Erstellung der Fehlermeldung verwendet, die in die Log-Datei eingefügt wird. Die Fehlermeldung hat ein ähnliches Format wie jene, die vom PHP-Interpreter erstellt wird.

PHP verfügt über die eingebaute Funktion `error_log()`, mit der sich eine Fehlermeldung in die Log-Datei schreiben lässt. Ihr einziger Parameter ist die gewünschte Fehlermeldung.

Da ein Fehler aufgetreten ist, wird die Funktion auch versuchen, den HTTP-Statuscode auf 500 zu setzen, um auf einen serverseitigen Fehler hinzuweisen. Dies gelingt mit der PHP-Funktion `http_response_code()`. (Sie muss dazu allerdings aufgerufen werden, bevor irgend etwas auf die Seite geschrieben wird.) Ihr einziger Parameter ist der zu verwendende Statuscode.

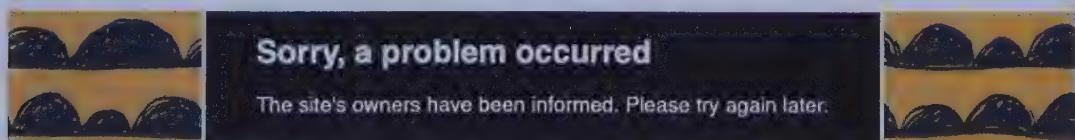
Bevor dem Besucher die Fehlermeldung angezeigt wird, bindet die Seite mit dem Befehl `require_once` die Header-Datei ein. So wird sichergestellt, dass die Fehlerseite das gleiche Design wie der Rest der Webseite hat. Mit dem `require_once`-Befehl anstelle von `include` wird sichergestellt, dass die Kopfzeile nur dann eingebunden wird, wenn dies auf der Seite nicht bereits zuvor geschehen ist.

Nach der Ausgabe einer benutzerfreundlichen Fehlermeldung wird mit dem Befehl `require_once` die Fußzeile der Seite eingebunden.

Abschließend sorgt der Befehl `exit` dafür, dass der PHP-Interpreter keinen weiteren Code auf der Seite ausführt.

```
<?php  
① set_error_handler('handle_error');  
  
② function handle_error($level, $message, $file = '', $line = 0)  
{  
③     $message = $level . ' ' . $message . ' in ' . $file . ' on line ' . $line;  
④     error_log($message);  
⑤     http_response_code(500);  
  
⑥     require_once 'includes/header.php';  
⑦     echo "<h1>Sorry, a problem occurred</h1>  
         The site's owners have been informed. Please try again later.";  
⑧     require_once 'includes/footer.php';  
⑨     exit;  
}  
⑩ $username = $_GET['username'];  
?>  
<?php include 'includes/header.php'; ?>  
<h1>Welcome, <?= $username ?></h1>  
<?php include 'includes/header.php'; ?>
```

ERGEBNIS



1. Die PHP-Funktion `set_error_handler()` weist den PHP-Interpreter an, im Falle eines nicht schwerwiegenden Fehlers die Funktion `handle_error()` aufzurufen (allerdings ohne die Klammern nach dem Funktionsnamen).
2. Die Funktion `handle_error()` wird definiert. Sie hat vier Parameter für die Daten, die der PHP-Interpreter an die Funktion übergibt: die Fehlerstufe, die Fehlermeldung, den Namen der Datei, in der der Fehler aufgetreten ist, und die Zeilennummer.
3. Mit den Informationen in den vier in Schritt 2 genannten Parametern wird eine Fehlermeldung erzeugt.
4. Mit der PHP-Funktion `error_log()` wird die Fehlermeldung in die Log-Datei von PHP geschrieben.
5. Die PHP-Funktion `http_response_code()` setzt den HTTP-Statuscode auf 500 und zeigt damit einen Serverfehler an.
6. Der Header wird eingebunden (falls noch nicht geschehen), um sicherzustellen, dass der Seitenkopf und die CSS-Stile eingefügt werden.
7. Eine benutzerfreundliche Fehlermeldung wird mit dem `echo`-Befehl auf der Seite ausgegeben.
8. Der Fußbereich wird eingefügt.
9. `exit` stoppt jeden weiteren Code.
10. Wenn die Seite versucht, auf einen nicht im superglobalen Array `$_GET` enthaltenen Schlüssel zuzugreifen, wird ein nicht schwerwiegender Fehler geworfen.

AUSNAHMEN (EXCEPTIONS)

Wenn ein Code normalerweise funktionieren würde, eine Ausnahmesituation dies jedoch verhindert, kann ein Exception-Objekt erstellt werden. Das gibt dem Code eine Chance, sich von Problemen zu erholen.

Wenn ein Exception-Objekt erzeugt wird, hält der PHP-Interpreter die Ausführung der Seite an und sucht nach Code zur Behandlung des außergewöhnlichen Zustands. Dies gibt dem Programm die Möglichkeit, sich von dem Problem zu erholen:

- Wenn es Code zur Behandlung der Situation findet, führt es diesen Code aus und setzt dann nach den Anweisungen, die die Ausnahme ausgelöst haben, die Code-Ausführung fort.

Findet er keinen Code zur Behandlung der Situation, gibt er einen schwerwiegenden Fehler aus, der mit der Meldung `Uncaught exception` beginnt, und die Seite wird nicht mehr weiter ausgeführt.

Programmierer sprechen davon, dass Exceptions geworfen werden und der Code zur Behandlung einer Exception die Exception abfängt.

Die Eigenschaften des Exception-Objekts speichern den Namen der Datei und die Zeile, in der das Problem aufgetreten ist, genau wie eine Fehlermeldung.

Wenn das Problem in einer Funktion oder Methode auftritt, enthält es außerdem einen sogenannten Stack-Trace, der aufzeichnet, welche Code-Zeilen diese Funktion oder Methode aufgerufen haben.

Der Stack-Trace kann sehr hilfreich sein, um die Ursache eines Problems zu finden, da mitunter verschiedene Seiten oder Code-Zeilen dieselbe Funktion oder Methode aufrufen. Und wenn ein Problem durch die Übergabe falscher Daten an eine Funktion oder Methode verursacht wird, ist es wichtig zu wissen, von wo aus der Aufruf dieser Funktion oder Methode erfolgte.

Exception-Objekte können auf zwei Arten erzeugt werden:

- Seit PHP 7 führen die meisten fatalen Fehler dazu, dass der PHP-Interpreter mit der eingebauten `Error-Klasse` ein Exception-Objekt erzeugt. Damit können sich Programme von einem schwerwiegenden Fehler erholen oder eine benutzerfreundliche Meldung anzeigen (anstatt dass die Seite abrupt abgebrochen wird).
- Programmierer können ein eigenes benutzerdefiniertes Exception-Objekt werfen, indem sie eine benutzerdefinierte Klasse verwenden, die auf der in PHP integrierten Exception-Klasse basiert.

Exceptions sollten nur dann verwendet werden, wenn der Programmierer weiß, dass der Code eigentlich funktionieren sollte, dies aber durch eine Ausnahmesituation verhindert wurde.

Eine Ausnahmesituation liegt dann vor, wenn Sie ein Problem vorhersehen, diesem aber nicht mittels Code vorbeugen können. Ein Beispiel:

- Datenbankgestützte Websites sind auf eine Datenbank angewiesen; wenn die Site keine Verbindung zur Datenbank herstellen kann, dann liegt eine Ausnahmesituation vor.
- Bei der Erfassung von Formulardaten geben Benutzer oft ungültige Werte ein; dies ist keine Ausnahmesituation und sollte im Code zur Datenvälidierung behandelt werden.

Der Code zur Behandlung der Ausnahme kann es der Website entweder ermöglichen, den Fehler zu beheben und den Betrieb fortzusetzen, oder dem Benutzer eine hilfreiche Meldung anzeigen.

Wenn eine Programmiererin eine Situation vorhersehen kann, die den Code möglicherweise an der Erfüllung seiner Aufgabe hindert (sie aber die Entstehung dieser Situation nicht beispielsweise durch Validierungs-Code verhindern kann), kann sie beim Eintreten dieser Situation ihr eigenes, selbst definiertes Exception-Objekt werfen. Damit lässt sich das Programm entweder wieder auffangen oder eine genaue Problembeschreibung aufzeichnen (zusammen mit einem Stack-Trace).

Zur Erstellung eines benutzerdefinierten Exception-Objekts sollten Sie zunächst eine benutzerdefinierte Exception-Klasse erstellen. Dies kann in einer einzigen Code-Zeile geschehen, da alle benutzerdefinierten Exceptions aus der integrierten Klasse `Exception` hervorgehen.

METHODE	RÜCKGABE
<code>getMessage()</code>	Eine Ausnahmemeldung. Bei Error-Exceptions ist dies die vom PHP-Interpreter erzeugte Fatal-Error-Meldung. Bei benutzerdefinierten Ausnahmen handelt es sich um eine vom Programmierer erstellte Meldung.
<code>getCode()</code>	Ein Ausnahme-Code, der zur Identifizierung des Ausnahmetyps verwendet wird. Bei Error-Exceptions wird dieser Code vom PHP-Interpreter generiert. Bei benutzerdefinierten Exceptions wird der Code vom Programmierer definiert.
<code>getFile()</code>	Der Name der Datei, in der die Ausnahme entstanden ist.
<code>getLine()</code>	Die Zeilennummer, in der die Ausnahme entstanden ist.
<code>getTraceAsString()</code>	Der Stack-Trace als Zeichenkette.
<code>getTrace()</code>	Der Stack-Trace als Array.

Zum Erstellen einer benutzerdefinierten Exception-Klasse verwenden Sie::

- Das Schlüsselwort `class`.
- Einen Klassennamen. Dieser Name beschreibt häufig den Zweck des Codes, in dem die Ausnahmesituation aufgetreten ist.
- Das Schlüsselwort `extends`, das anzeigt, dass diese Klasse eine andere bestehende Klasse erweitert.
- Den Namen der zu erweiternden Klasse (in diesem Fall die integrierte Klasse `Exception`). Die neue Klasse erbt also die Eigenschaften und Methoden der Klasse, die sie erweitert.
- Ein Paar geschweifte Klammern.

Wenn eine Klasse eine andere Klasse erweitert, erbt sie deren Eigenschaften und Methoden. Benutzerdefinierte Exception-Klassen verfügen daher über alle Eigenschaften und Methoden, die auch in der integrierten Klasse `Exception` definiert sind.

Sowohl die `Exception`-Klasse als auch die integrierte `Error`-Klasse verwenden die Schnittstelle `Throwable`. Eine Schnittstelle beschreibt die Namen der Eigenschaften und/oder Methoden, die von einem Objekt implementiert werden, sowie die Daten, die sie zurückgeben sollen. Die folgende Tabelle zeigt die Methoden der `Throwable`-Schnittstelle. Alle `Exception`-Objekte verfügen über diese Methoden.

Verwenden Sie das Schlüsselwort `throw`, um eine Exception mit einer von Ihnen definierten Exception-Klasse zu erzeugen oder auszulösen:

- Das Schlüsselwort `throw` (damit wird nicht nur das Exception-Objekt erzeugt, sondern der PHP-Interpreter zudem angewiesen, nach Code zu suchen, der die Exception abfängt, wenn sie erzeugt wird)
- Das Schlüsselwort `new` (um ein neues Objekt zu erzeugen)
- Den Namen einer eigenen Exception-Klasse

Fügen Sie dann in Klammern hinzu:

- Eine Fehlermeldung, die das Problem beschreibt
- Optionalen Code zur Identifizierung des Problems

```
NAME DER BENUTZERDEFINIERTEN EXCEPTION-KLASSE
class CustomExceptionName extends Exception {};
throw new CustomExceptionName($message[, $code]);
```

NAME DER EXCEPTION-KLASSE NACHRICHT CODE

AUSNAHMEN MIT TRY ... CATCH BEHANDELN

Wenn Sie wissen, dass bestimmter Code in Ausnahmesituationen zu einer Exception führen kann und Sie das Problem bereinigen können, lässt sich diese Situation mit einer try... catch-Anweisung handhaben.

In einer try... catch-Anweisung folgt auf das Schlüsselwort `try` ein Code-Block, dessen Anweisungen der PHP-Interpreter versuchen soll, auszuführen, wobei jedoch eine Ausnahme auftreten könnte. Wenn im `try`-Block eine Exception ausgelöst wird, wird der PHP-Interpreter:

- die Code-Ausführung anhalten
- nach dem `catch`-Block suchen, der anschließend folgen sollte
- prüften, ob dieser `catch`-Block die Situation bewältigen kann (weil er die Klasse benennt, mit der das Exception-Objekt erstellt wurde, oder eine davon implementierte Schnittstelle)
- wenn ja, werden die Anweisungen in diesem `catch`-Block ausgeführt

Geben Sie in den Klammern nach dem Schlüsselwort `catch` an:

- die Klasse, mit der das Exception-Objekt erstellt wurde, oder eine davon implementierte Schnittstelle (auf den Seiten 374 und 375 sehen Sie, wie Sie mehrere `catch`-Blöcke angeben können, von denen jeder in der Lage ist, Ausnahmen zu behandeln, die mit verschiedenen Klassen erstellt wurden)
- den Namen einer Variablen, die das Exception-Objekt im `catch`-Block aufnimmt (sie wird oft `$e` genannt)

```
try {  
    // versuche, etwas zu tun, das eine Ausnahme auslösen könnte  
} catch (ExceptionClassName $e) {  
    // falls ausgelöst, tue etwas, um diese Ausnahme zu behandeln  
} finally () {  
    // tue in jedem Fall etwas, egal ob die Ausnahme ausgelöst wurde oder nicht  
}
```

Wenn im `try`-Block keine Ausnahme ausgelöst wird, überspringt der PHP-Interpreter den `catch`-Block.

Auf den `catch`-Block kann auch ein optionaler `finally`-Block folgen. Die Anweisungen im `finally`-Block werden immer ausgeführt, egal ob eine Ausnahme ausgelöst wurde oder nicht.

Sobald eine Ausnahme behandelt wurde, führt der PHP-Interpreter die Code-Zeile nach dem `catch`-Block aus. Er tut dies, ohne Details über den Auslöser der Ausnahme in die PHP Log-Datei zu schreiben. Das bedeutet, dass die Programmierer nicht wissen, wie oft die Ausnahmesituation aufgetreten ist.

Wenn Sie Details zu einer behandelten Ausnahme im Fehlerprotokoll aufzeichnen möchten, können Sie die PHP-Funktion `error_log()` verwenden. Sie braucht nur ein Argument: das ausgelöste Exception-Objekt. Der PHP-Interpreter nimmt die im Exception-Objekt gespeicherten Daten über das Problem, wandelt sie in eine Zeichenkette um und fügt sie dann der Log-Datei hinzu.

STANDARDFUNKTION ZUR BEHANDLUNG VON AUSNAHMEN

Der PHP-Interpreter kann eine benutzerdefinierte Funktion ausführen, um Ausnahmen zu behandeln, die nicht von einem catch-Block behandelt wurden (weil sie nicht in einem try-Block geworfen wurden oder weil sie nicht die vom catch-Block angegebene Klasse verwenden).

Programmierer können den Namen einer benutzerdefinierten Funktion angeben, die als Default Exception Handler bezeichnet wird und die der PHP-Interpreter aufrufen soll, wenn eine Ausnahme nicht durch einen catch-Block behandelt wird.

Die PHP-Funktion `set_exception_handler()` gibt den Namen einer benutzerdefinierten Funktion an, die aufgerufen werden soll. Ihr einziger Parameter ist der Funktionsname. Auf den Namen sollten keine Klammern folgen.

```
set_exception_handler('name')
  └─
    FUNKTIONSNAME
```

Die grundlegende Funktion zur Behandlung von Exceptions unten:

- fügt der Log-Datei Einzelheiten über das Problem hinzu
- setzt den korrekten HTTP-Statuscode (500)
- zeigt eine Meldung für den Benutzer an
- stoppt die Ausführung von weiterem Code auf der Seite

Ein Beispiel für eine Standardfunktion zur Behandlung von Ausnahmen finden Sie auf den Seiten 376–377.

Eine kompliziertere Funktion zur Behandlung von Ausnahmen könnte die Klasse abfragen, mit der die Exception erzeugt wurde, und auf jede Ausnahme auf eine andere Weise reagieren.

```
function handle_exception($e)
{
    error_log($e);
    http_response_code(500);
    echo '<h1>Sorry, an error occurred please try again later.</h1>';
    exit;
}
```

AUSNAHMEN MIT TRY ... CATCH BEHANDELN

1. In diesem Beispiel enthält der try-Block Code, der eine Ausnahme auslösen kann.

2. Stellen Sie sich vor, dass die Include-Datei Code zur Anzeige von Werbung enthält, was normalerweise auch funktioniert, aber dass ein Problem in diesem Code gelegentlich zu einer Exception führt.

3. Wenn im try-Block eine Exception ausgelöst wird, sucht der PHP-Interpreter nach einem catch-Block zur Behandlung der Exception.

Auf das Schlüsselwort `catch` folgt ein Klammerpaar. Darin stehen:

- Der Klassename, der darauf hindeutet, dass dieser catch-Block für jedes Exception-Objekt ausgeführt wird, das mit der Klasse `Exception` erstellt wurde.
- `$e`, der Name einer Variablen, in der das Exception-Objekt gespeichert wird, damit seine Daten im catch-Block genutzt werden können.

4. Eine Platzhalteranzeige wird eingeblendet. Das ist besser, als wenn die Seite beim Auftreten der Ausnahme nicht mehr weiterläuft.

5. Die PHP-Funktion `error_log()` fügt den Fehler in die PHP-Log-Datei ein. Ihr einziges Argument ist das Exception-Objekt, das in der Include-Datei erstellt wurde.

section_b/c10/live/try-catch.php

PHP

```
<?php include 'includes/header.php'; ?>

<?php
① try {
②     include 'includes/ad-server.php';
③ } catch (Exception $e) {
④     echo '';
⑤     error_log($e);
}
?>
<h1>Latest Products</h1>
...
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS



Hinweis: Für dieses Beispiel löst die Include-Datei `ad-server.php` im Download-Code eine Ausnahme aus, um sicherzustellen, dass der catch-Block ausgeführt wird.

Auf Seite 374 erfahren Sie, wie Sie verschiedenen Code in einer Reihe von catch-Blöcken unterbringen, um mit verschiedenen Klassen erstellte Exception-Objekte zu behandeln.

BENUTZERDEFINIERT EXCEPTIONS WERFEN

Die nachstehende Klasse ImageHandler manipuliert Bilder mit GD. Bei falscher Verwendung löst sie benutzerdefinierte Ausnahmen aus (auf der nächsten Seite wird die Verwendung der Klasse beschrieben).

1. Eine eigene Exception-Klasse ImageHandlerException wird erstellt. Sie erbt die Eigenschaften und Methoden der PHP-eigenen Exception-Klasse.

2. Beim Erstellen eines Objekts mit der Klasse ImageHandler prüft die Methode __construct(), ob das Bild einem der zulässigen Medientypen entspricht. Ist dies nicht der Fall, wird mithilfe der Klasse ImageHandlerException eine Ausnahme ausgelöst.

Die Fehlermeldung zeigt an, dass das Bildformat nicht akzeptiert wird, und der Fehlercode 1 wird vergeben.

3. Beim Aufruf der Methode resizeImage() wird eine Ausnahme ausgelöst, wenn der Benutzer versucht, ein Bild zu erstellen, das größer ist als das von ihm hochgeladene Originalbild, da dies die Bildqualität beeinträchtigt.

Die Fehlermeldung besagt, dass das Originalbild zu klein ist, und der Fehlercode lautet 2.

PHP

section_b/c10/live/classes/ImageHandler.php

```
<?php  
① class ImageHandlerException extends Exception {};  
  
class ImageHandler  
{  
    public $fileTypes = ['image/jpeg', 'image/png',]; // erlaubte Medientypen  
    ...  
    public function __construct(string $filepath, string $filename)  
    {  
        ...  
        ② if (!in_array($this->mediaType, $this->fileTypes)) { // wenn Medientyp nicht erlaubt  
            throw new ImageHandlerException('File not an accepted image format', 1);  
        }  
        ...  
    }  
    public function resizeImage(int $newWidth, int $newHeight, string $uploadPath)  
    {  
        if (($this->origWidth < $newWidth)  
        or ($this->origHeight < $newHeight)) // wenn Originalbild zu klein  
            throw new ImageHandlerException('Original image too small', 2);  
        }  
        // Code zum Skalieren und Speichern des Bilds hierher ...  
    }  
}
```

VERSCHIEDENE ARTEN VON EXCEPTIONS ABFANGEN

Auf dieser Seite können sich Nutzer durch Angabe ihrer E-Mail-Adresse und Hochladen eines Profilbilds für eine Website registrieren. Normalerweise funktioniert das gut, aber in Ausnahmefällen kann das Bild nicht gespeichert werden. In solchen Fällen wird die Ausnahme behandelt, damit der Code weiterlaufen und die E-Mail-Adresse des Besuchers speichern kann (auch wenn sich das Bild nicht speichern lässt). Das Beispiel verwendet zwei catch-Blöcke, um verschiedene Arten von Ausnahmen zu behandeln, die beim Hochladen eines Bilds auftreten können.

1. Die Datei `ImageHandler.php` von der vorherigen Seite wird eingebunden. Sie enthält die Klassen `ImageHandlerException` sowie `ImageHandler`. Letztere dient zur Größenänderung und Speicherung hochgeladener Bilder.
2. Ein try-Block enthält Anweisungen zur Erstellung eines `ImageHandler`-Objekts für das vom Benutzer hochgeladene Profilbild, zur Größenanpassung und Speicherung des Bilds und zur dessen anschließender Anzeige. Es folgen zwei catch-Blöcke.
3. Die Klammern des ersten catch-Blocks enthalten den Namen der Klasse `ImageHandlerException`. Das bedeutet, dass der PHP-Interpreter diesen catch-Block ausführt, wenn der Code im try-Block unter Verwendung der Klasse `ImageHandlerException` ein Exception-Objekt erzeugt hat. Im catch-Block wird das Exception-Objekt in der Variablen `$e` gespeichert.
4. Die von der Klasse `ImageHandler` erzeugten Ausnahmen sind benutzerfreundlich, sodass die Methode `getMessage()` des Exception-Objekts (das es von der eingebauten Exception-Klasse geerbt hat) die Fehlermeldung übernimmt und sie in der Variablen `$message` speichert. Die Nachricht teilt dem Benutzer entweder mit, dass das Bild den falschen Medientyp hatte oder zu klein war.

Falls der erste catch-Block die Ausnahme behandelt hat, fährt der PHP-Interpreter mit dem Rest der Seite fort und überspringt alle weiteren catch-Blöcke.

5. Falls der erste catch-Block nicht durchlaufen wurde (weil die im try-Block ausgelöste Ausnahme nicht mit der Klasse `ImageHandler` erstellt wurde), wird der zweite catch-Block ausgeführt.

Die Klammern des zweiten catch-Blocks verweisen auf die Throwable-Schnittstelle und besagen, dass er jede Ausnahme abfangen soll, die die Throwable-Schnittstelle implementiert. Er behandelt also jede bisher noch un behandelte Ausnahme aus dem try-Block. Dazu gehören auch schwerwiegende PHP-Fehler, die der PHP-Interpreter möglicherweise ausgelöst hat (z. B. wenn das Bild nicht gespeichert werden konnte, weil die Festplatte voll war).

Normalerweise sollten catch-Blöcke versuchen, bestimmte Ausnahmeklassen abzufangen, anstatt wie hier alle Ausnahmen abzufangen. Dies könnte jedoch auch gerechtfertigt sein, wenn:

- die Ausnahme eine entscheidende Aktion verhindern könnte, z. B. die Registrierung eines Benutzers (es ist besser, die E-Mail-Adresse des Benutzers zu haben als gar nichts)
 - es sich um eine vorübergehende Maßnahme handelt, während die genaue Ursache des Problems ermittelt wird
6. Dieser catch-Block geht anders mit der Situation um. In der Variablen `$message` wird eine Meldung abgelegt, die darüber informiert, dass das Bild nicht gespeichert wurde. Die Fehlermeldung des Exception-Objekts wird nicht angezeigt, da diese Informationen enthalten könnte, die den Anwender verwirren oder Hackern sensible Informationen liefern könnten.
 7. Die PHP-Funktion `error_log()` wird aufgerufen, um Informationen über die Ausnahme in das PHP-Fehlerprotokoll aufzunehmen.

Sobald der Code in diesem catch-Block ausgeführt wurde, wird der restliche Code der Seite abgearbeitet. Dieser könnte z. B. die E-Mail-Adresse des Besuchers in einer Datenbank speichern. Wäre die Ausnahme nicht behandelt worden, könnten auch keine Benutzer- eingaben gespeichert werden.

```
<?php  
① include 'classes/ImageHandler.php';           // Klasse einbinden  
$message = '';  
$thumb   = '';  
$email   = '';  
  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {        // wenn Formular abgesendet  
    $email = $_POST['email'] ?? '';  
    if ($_FILES['image']['error'] == 0) {  
        $file = $_FILES['image']['name'];  
        $temp = $_FILES['image']['tmp_name'];  
  
        [ ② try {  
            $image = new ImageHandler($temp, $file); // Objekt erstellen  
            $thumb = $image->resizeImage(300, 300, 'uploads/'); // Bild skalieren  
            $message = ''; // Bild in $message speichern  
        } catch (ImageHandlerException $e) {          // wenn ImageHandlerException  
            $message = $e->getMessage();             // Fehlermeldung abrufen  
        } catch (Throwable $e) {                      // wenn anderer Grund  
            $message = 'We were unable to save your image'; // generische Fehlermeldung  
            error_log($e);                          // Fehler protokollieren  
        }  
    }  
    // hier könnte die Seite die E-Mail-Adresse speichern  
}  
?  
<?php include 'includes/header.php' ?>  
<h1>Join Us</h1>  
<?= $message ?>  
...
```

ERGEBNIS



STANDARDMÄSSIGE FEHLER- UND AUSNAHMEBEHANDLUNG

Dieses Beispiel zeigt, wie Sie jeden Fehler und jede unbehandelte Ausnahme auf eine beständige Weise behandeln können.

Zunächst legt die PHP-Funktion

`set_exception_handler()` fest, dass die benutzerdefinierte Funktion `handle_exception()` aufgerufen werden soll, wenn eine Ausnahme ausgelöst wird, die nicht durch einen catch-Block behandelt wird.

Diese Funktion:

- protokolliert das Problem
- legt den HTTP-Statuscode fest
- zeigt eine verständliche Fehlermeldung an
- stoppt die Code-Ausführung

Als Nächstes legt die PHP-Funktion

`set_error_handler()` fest, dass beim Auftreten eines nicht schwerwiegenden Fehlers die Funktion `error_handler()` aufgerufen werden soll.

NICHT ABGEFANGENE EXCEPTIONS

1. Die PHP-Funktion `set_exception_handler()` legt eine benutzerdefinierte Funktion fest, die aufgerufen wird, wenn eine Ausnahme ausgelöst wird und nicht durch einen catch-Block behandelt wird.

2. Die Funktion `handle_exception()` wird definiert. Ihr einziger Parameter ist das Exception-Objekt, das geworfen wurde.

3. Die Ausnahme wird in der Log-Datei von PHP protokolliert.

4. Der HTTP-Statuscode wird auf 500 gesetzt.

5. Falls noch nicht geschehen, wird die Kopfzeile eingebunden.

6. Eine benutzerfreundliche Fehlermeldung wird angezeigt.

7. Falls noch nicht geschehen, wird die Fußzeile eingebunden.

Danach führt der Interpreter keinen weiteren Code mehr aus.

`error_handler()` wandelt nicht schwerwiegende Fehler in Exceptions um, sodass sie auf die gleiche Weise behandelt werden wie schwerwiegende Fehler (die Exceptions auslösen).

Schließlich legt die PHP-Funktion `register_shutdown_function()` fest, dass die benutzerdefinierte Funktion `handle_shutdown()` aufgerufen werden soll, wenn eine Seite nicht mehr läuft. Diese Funktion wird aufgerufen, wenn ein schwerwiegender Fehler nicht in eine Error-Exception umgewandelt wird oder vom Default-Error-Handler nicht behandelt werden kann.

Die Funktion `handle_shutdown()` prüft mithilfe der PHP-Funktion `error_get_last()`, ob beim Ausführen der Seite ein Fehler aufgetreten ist. Wenn dies der Fall ist, wird der Fehler in eine Ausnahme umgewandelt, und die Exception-Handler-Funktion wird aufgerufen, um den Fehler zu behandeln.

NICHT SCHWERWIEGENDE FEHLER

9. Die Funktion `set_error_handler()` legt eine benutzerdefinierte Funktion fest, die aufgerufen wird, wenn der PHP-Interpreter einen nicht schwerwiegenden Fehler meldet.

10. Dem Default-Error-Handler werden Informationen über den Fehler (Typ und Meldung sowie die Datei und die Zeile seines Auftretens) übergeben, wie auf Seite 366 beschrieben.

11. Aus den Fehlerdaten wird eine neue Exception erstellt. Auf diese Weise kann die Site alle nicht schwerwiegenden Fehler auf die gleiche Weise behandeln wie schwerwiegende Fehler und Ausnahmen (mit der gleichen `handle_exception()`-Funktion).

Das Exception-Objekt wird mithilfe der in PHP eingebauten `ErrorException`-Klasse geworfen, die in PHP aufgenommen wurde, damit Fehler in Exceptions umgewandelt werden können. Der zweite Parameter ist ein optionaler Fehlercode (eine ganze Zahl), der die Exception kennzeichnet (hier ist er 0).

```

<?php ...
① set_exception_handler('handle_exception');           // Exception-Handler setzen
② function handle_exception($e)
{
    ③ error_log($e);                                // Fehler protokollieren
    ④ http_response_code(500);                      // Statuscode setzen
    ⑤ require_once 'header.php';                     // Kopfzeile sicher einbinden
    ⑥ [ echo "<h1>Sorry, a problem occurred</h1>
        <p>The site's owners have been informed. Please try again later.</p>";
    ⑦ require_once 'footer.php';                     // Fußzeile einbinden
    ⑧ exit;                                         // Code-Ausführung stoppen
}

⑨ set_error_handler('handle_error');                 // Error-Handler setzen
⑩ function handle_error($type, $message, $file = '', $line = 0)
{
    ⑪ throw new ErrorException($message, 0, $type, $file, $line); // ErrorException werfen
}

⑫ register_shutdown_function('handle_shutdown');     // Shutdown-Handler setzen
⑬ function handle_shutdown()
{
    ⑭ $error = error_get_last();                     // Gab es einen Fehler im Skriptablauf?
    ⑮ if ($error) {                                 // wenn ja, Exception auslösen
        ⑯ [ $e = new ErrorException($error['message'], 0, $error['type'],
                                    $error['file'], $error['line']);
    ⑰ handle_exception($e);                         // Exception-Handler aufrufen
}
}

```

SCHWERWIEGENDE FEHLER

12. Die PHP-Funktion

`register_shutdown_function()` weist den PHP-Interpreter an, die benutzerdefinierte Funktion `handle_shutdown()` aufzurufen, wenn eine Seite nicht mehr läuft. Dies geschieht, weil einige schwerwiegende Fehler nicht in Ausnahmen umgewandelt und vom Default-Error-Handler nicht behandelt werden.

13. Die Funktion `handle_shutdown()` wird definiert.

`14. Die PHP-Funktion error_get_last() prüft, ob bei der Ausführung der Seite ein Fehler aufgetreten ist. Wenn ja, werden die Details des letzten Fehlers in einem Array zurückgegeben; wenn nicht, wird null zurückgegeben. Der Rückgabewert wird in $error gespeichert.`

15. Eine `if`-Anweisung prüft, ob die Variable `$error` einen Wert hat, der auf einen Fehler hinweist.

16. Wenn ja, wird der Fehler in eine Ausnahme umgewandelt. Hinweis: Das Schlüsselwort `throw` wird hier nicht verwendet, da der Default-Exception-Handler keine Ausnahmen abfängt, die in einer Shutdown-Funktion ausgelöst werden. Stattdessen wird das Objekt wie jedes andere Objekt erstellt.

17. `handle_exception()` wird aufgerufen. Das in Schritt 16 erstellte Exception-Objekt wird als Argument übergeben.

Probieren Sie es: `example.php` im Download-Code hat auskommentierte Zeilen, um die Fehler- und Ausnahmebehandlungen zu testen. Heben Sie die Auskommentierungen nacheinander einzeln auf.

WEBSERVER-FEHLER ANZEIGEN

Wenn ein Webserver die angeforderte Datei nicht finden kann oder der Server die Anfrage des Browsers wegen eines internen Fehlers nicht bearbeiten kann, sendet der Webserver einen Fehlercode und eine Webseite zurück an den Browser.

Auf den Seiten 180 und 181 haben Sie gesehen, wie Server HTTP-Header zusammen mit der angeforderten Datei an den Browser senden. Einer dieser Header ist ein Statuscode, der angibt, ob die Anfrage erfolgreich war oder nicht.

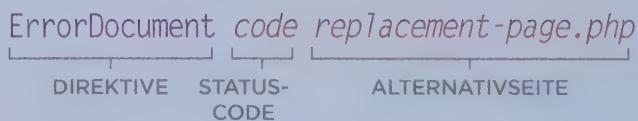
- Wenn ein Server die Anfrage erfolgreich bearbeitet hat, gibt er zusammen mit der angeforderten Datei den Code 200 zurück.
- Wenn die Datei nicht gefunden werden kann, wird der Code 404 zusammen mit einer Fehlerseite zurückgegeben, die das Problem beschreibt.
- Wenn ein Serverfehler die Anzeige einer Seite verhindert, wird der Code 500 und eine Fehlerseite mit einem Hinweis auf einen internen Serverfehler zurückgegeben.

Die vom Webserver erstellten Fehlerseiten sind nicht sehr benutzerfreundlich. Sie können aber auch Ihre eigenen benutzerdefinierten Fehlerseiten erstellen, die der Webserver dann statt seiner eigenen senden kann.

Mit solchen benutzerdefinierten Fehlerseiten können Sie den Besuchern eine klarere Problembeschreibung liefern und dabei auch die gleiche Optik wie bei den übrigen Seiten der Website verwenden.

Um eine benutzerdefinierte Fehlerseite zu senden, fügen Sie eine ErrorDocument-Direktive in die .htaccess-Datei ein und geben dabei Folgendes an:

- den Statuscode, für den diese Datei verwendet werden soll
- den Pfad zur Datei, die für diesen Code angezeigt werden soll



Die Standard-Fehlerseite von Apache, wenn eine Datei nicht gefunden wird:

Not Found

The requested URL /phpbook/section_b/c10/missing.php was not found on this server.

Die Standard-Fehlerseite von Apache, wenn ein interner Fehler die Anzeige einer Seite verhindert:

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

PHP

① ErrorDocument 404 /code/section_b/c10/live/page-not-found.php
 ② ErrorDocument 500 /code/section_b/c10/live/error.php

PHP

section_b/c10/live/page-not-found.php

```
<?php require_once 'includes/header.php'; ?>
<h1>Sorry! We cannot find that page.</h1>
②<p>Try the <a href="index.php">home page</a> or email us at
  <a href="mailto:hello@eg.link">hello@eg.link</a>.</p>
<?php require_once 'includes/footer.php'; ?>
```

PHP

section_b/c10/live/error.php

```
<?php include 'includes/header.php'; ?>
③<h1>Sorry! An error occurred.</h1>
<p>The site owners have been informed. Please try again soon.</p>
<?php include 'includes/footer.php'; ?>
```

ERGEBNIS**ERGEBNIS**

1. Die Datei .htaccess verwendet die Anweisung ErrorDocument, um Pfade zu benutzerdefinierten Fehlerseiten anzugeben, die angezeigt werden sollen, wenn:

a) eine Datei nicht gefunden werden kann

b) ein Serverfehler vorliegt

2. Die Datei page-not-found.php erklärt auf benutzerfreundliche Weise, dass die Datei nicht gefunden werden konnte.

3. Die Datei error.php teilt den Besuchern mit, dass ein Fehler aufgetreten ist.

Probieren Sie es: Fordern Sie im Live-Verzeichnis für dieses Kapitel eine nicht existente Seite an, z. B. missing.php.

Hinweis: Fehlerseiten sollten keinen Code enthalten, der eine Datenbankverbindung aufbaut, da die Fehlerseite dann bei einem Datenbankproblem nicht angezeigt werden würde.

ZUSAMMENFASSUNG

FEHLERBEHANDLUNG

- Fehlermeldungen helfen, das Problem zu erkennen.
- Geben Sie Fehler in der Entwicklungsphase einer Website auf dem Bildschirm aus.
- Wenn eine Website live geht, zeigen Sie Fehler nicht mehr auf dem Bildschirm an. Schreiben Sie sie in eine Log-Datei.
- Beim Auftreten eines nicht schwerwiegenden Fehlers kann eine Fehlerbehandlungsfunktion ausgeführt werden.
- Exceptions sind Objekte, die beim Auftreten außergewöhnlicher Umstände erzeugt werden, die die normale Ausführung einer Seite verhindern.
- Wenn ein Exception-Objekt erzeugt (oder geworfen) wird, sucht der Interpreter nach einem alternativen Code-Block, den er ausführen kann.
- Exceptions können mit einer `try... catch`-Anweisung oder einer Ausnahmebehandlungsfunktion abgefangen werden.
- Fehler können in Ausnahmen umgewandelt werden, sodass alle Probleme auf dieselbe Weise behandelt werden.

C

DATENBANK- GESTÜTZTE WEBSITES

Datenbankgestützte Websites verwenden eine Datenbank, um den Großteil der auf den Seiten angezeigten Inhalte sowie weitere Daten wie etwa Informationen über die registrierten Mitglieder zu speichern.

Da PHP auf die Daten in einer Datenbank zugreifen und diese aktualisieren kann, können datenbankgestützte Websites:

- technisch weniger versierten Benutzern erlauben, Website-Inhalte mithilfe von Formularen auf einer Webseite zu erstellen oder zu aktualisieren (sie brauchen keine Kenntnisse im Schreiben von Code oder im Umgang mit FTP zur Aktualisierung von Dateien auf dem Server).
- mit Daten aus der Datenbank Seiten erstellen, die auf jedes einzelne Mitglied der Website zugeschnitten sind.

Dieses Buch verwendet zur Erstellung und Verwaltung der Website-Datenbank die Software **MySQL**. Die Datenbank speichert Daten in einer Reihe von Tabellen (jede Tabelle ist mit einer Tabellenkalkulation vergleichbar). Da sich die Daten in einer Tabelle oft auf Daten in einer anderen Tabelle beziehen, wird sie als **relationale Datenbank** bezeichnet.

(Immer, wenn wir uns auf MySQL beziehen, gelten die Informationen auch für das auf Seite 15 vorgestellte MariaDB.)

PHP verfügt über eine Reihe integrierter Klassen, die sogenannten **PHP Data Objects (PDO)**, mit denen Sie auf die Daten in der Datenbank zugreifen und sie aktualisieren können.

Um datenbankgestützte Websites zu erstellen, müssen Sie also lernen:

- dass eine Sprache namens **SQL (Structured Query Language**, ausgesprochen: sequel oder ess-quell) verwendet wird, um Daten von einer Datenbank anzufordern und die darin enthaltenen Daten zu aktualisieren.
- wie Sie PDO zur Ausführung von SQL-Befehlen verwenden, die Daten aus einer Datenbank abfragen und diese Daten Ihrem PHP-Code zur Verfügung stellen.
- wie Sie PDO zur Ausführung von SQL-Befehlen verwenden, die die in einer Datenbank gespeicherten Daten aktualisieren.

MySQL besitzt selbst keine grafische Benutzeroberfläche, aber mit dem kostenlosen Tool **phpMyAdmin** können Sie die Datenbank verwalten und ihren Inhalt einsehen. Mehr zu seiner Verwendung erfahren Sie in der Einleitung zu diesem Abschnitt.

Bevor Sie die Kapitel in diesem Abschnitt lesen, müssen Sie verstehen, wie Datenbanken die für eine Website benötigten Daten speichern und wie Sie eine Datenbank mit phpMyAdmin verwalten.

DIE BEISPIEL-WEBSITE

In der zweiten Buchhälfte wird eine Beispielanwendung entwickelt. Sie demonstriert, wie datenbankgestützte Websites erstellt werden und noch weitere Konzepte, die Sie lernen werden. Zu Beginn stellen wir Ihnen die Beispiel-Website vor, damit Sie die zugehörige Beispieldatenbank besser verstehen.

WIE DATENBANKEN DATEN SPEICHERN

Als Nächstes erfahren Sie, wie Datenbanken Daten speichern. Die Daten sind dabei in mehreren Tabellen strukturiert. Sie lernen auch die Datentypen von MySQL kennen (die sich von den PHP-Datentypen unterscheiden).

SO VERWENDEN SIE PHPMYADMIN

phpMyAdmin ist ein Tool, das auf einem Webserver läuft. Es ist eine Art Website zur Verwaltung einer MySQL-Datenbank. Sie lernen, wie Sie damit:

- neue Datenbanken erstellen.
- Daten in Datenbanken einsehen.
- Backups von Datenbanken erstellen.

WIE SIE EINE DATENBANK ERSTELLEN

Als Nächstes erfahren Sie, wie Sie die Datenbank für die Beispielanwendung einrichten, die im weiteren Verlauf des Buchs entwickelt wird. Die Datenbank enthält den Inhalt der Website und Daten über ihre Mitglieder.

Zunächst lernen Sie, eine leere Datenbank zu erstellen. Dann führen Sie etwas SQL-Code (aus dem Code-Download) aus, der:

- die Tabellen für die Beispieldatenbank erstellt.
- Daten in jede dieser Tabellen einfügt.

Hinweis: Alle weiteren Beispiele stützen sich auf diese Datenbank, sodass sie erstellt und mit Daten gefüllt werden muss, um fortfahren zu können.

BENUTZERKONTEN FÜR DIE DATENBANK ERSTELLEN

Schließlich lernen Sie, Benutzerkonten für die Datenbank einzurichten. Der PHP-Code braucht ein Benutzerkonto, um sich mit der Datenbank zu verbinden (so wie ein E-Mail-Programm ein E-Mail-Konto zum Versand und Empfang von E-Mails benötigt).

VORSTELLUNG DER BEISPIEL-WEBSITE

Die Beispiel-Website, die wir im weiteren Verlauf dieses Buchs aufbauen, ist ein einfaches **Content-Management-System** (CMS), also ein Werkzeug, um den Inhalt einer Website zu pflegen, ohne dafür Code schreiben zu müssen.

Das im weiteren Verlauf dieses Buchs entstehende Content-Management-System stellt die Arbeit eines Kreativkollektivs vor, aber es könnte für viele verschiedene Websites als CMS genutzt werden.

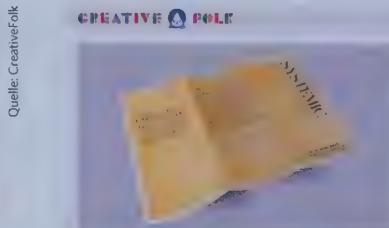
Im oberen Bereich jeder Seite zeigt die Navigationsleiste nach dem Website-Namen (Creative Folk) ihre Bereiche oder **Kategorien** an.

Die Startseite (unten) besteht aus der Datei index.php. Unter der Navigationsleiste erscheinen Informationen über die sechs neuesten Artikel. Wenn ein neuer Artikel hochgeladen wird, erscheint er auf der Startseite, und der älteste zuvor dargestellte Artikel wird dann nicht mehr angezeigt.

Jeder **Artikel** stellt eine kreative Arbeit vor. Alle Artikel werden über dieselbe Datei article.php angezeigt, deren Aufgabe es ist, jeweils einen Artikel aus der Datenbank abzurufen und diese Daten in die Seite einzufügen.

Die Artikelseite zeigt ein Bild des Werks, seinen Titel, das Erscheinungsdatum des Artikels auf der Website, eine Beschreibung des Werks, die zugehörige Kategorie und den Namen der Person, die es erstellt hat.

Jeder Artikel verfügt außerdem über die Option 'published'; Der Artikel wird erst dann veröffentlicht, wenn diese Option ausgewählt wurde.



In späteren Kapiteln wird die Website weiterentwickelt, um den Betrachtern die Möglichkeit zu bieten, eigene Werke hochzuladen, »Likes« zu vergeben und Artikel zu kommentieren.

Die Artikel sind in **Kategorien** gruppiert (die wie Abschnitte der Website sind). Alle Kategorien werden auf der gleichen Seite category .php angezeigt. Sie zeigt den Namen und die Beschreibung der Kategorie an, gefolgt von einer Zusammenfassung der einzelnen Artikel in der Kategorie.

Für jeden Artikel werden ein Bild und sein Titel, eine kurze Beschreibung des Werks, die Kategorie, in der er sich befindet, und sein Autor angezeigt (dies sind die gleichen Daten, die für die letzten sechs Artikel auf der Startseite angezeigt werden).

Für jede Kategorie können Sie außerdem festlegen, ob ihr Name in der Hauptnavigationsleiste erscheinen oder ausgeblendet werden soll.

Jeder Artikel wird von einem **Mitglied** der Website geschrieben; das Mitglied ist der **Autor** des Artikels. Das Profil jedes Mitglieds wird über dieselbe Datei member .php angezeigt.

Es beinhaltet den Namen des Autors, das Datum seiner Registrierung auf der Website und sein Profilbild, gefolgt von einer Liste seiner Artikel. Die Informationen zu jedem Artikel stimmen mit den Informationen überein, die auf der Kategorie- und der Startseite angezeigt werden (Bild, Titel, Kurzbeschreibung, Kategorienname und Autor).

Am Ende dieses Buchs können auch unregistrierte Besucher ihre eigenen Arbeiten einreichen, markieren, welche Artikel ihnen gefallen, und Kommentare zu den Artikeln hinterlassen.

Creative Folk

Print Digital Illustration Photography ↗

PRINT



Travel Guide

Book design for stores of travel guides



Polite Society Posters

Poster designs for a fast-food chain



Chimney Business Cards

Business cards for chimney sweepers



Milk Beach Album Cover

Book design for music series



The Ios Palace

Book design for travel guide



Systemic Brochure

Book design for systematic

Quelle: CreativeFolk

Creative Folk

Print Digital Illustration Photography ↗

IVY STONE

MEMBER SINCE 2012



Travel Guide

Book design for stores of travel guides



Polite Society Posters

Poster designs for a fashion store



Floral Website

Book design for floral website



Polite Society Website



Milk Beach Album Cover



Polite Society Mural

Quelle: CreativeFolk

WIE EINE RELATIONALE DATENBANK DATEN SPEICHERT

Relationale Datenbanken speichern Daten in Tabellen. Eine einzelne Datenbank kann aus vielen Tabellen bestehen. Unten sehen Sie die Datenbanktabellen für die Beispiel-Website, die im weiteren Verlauf dieses Buchs entsteht.

Ein **relationales Datenbankmanagementsystem (RDBMS)** wie MySQL oder MariaDB ist eine Software, die mehrere Datenbanken verwalten kann (so wie ein Webserver mehrere Websites hosten kann).

Die Datenbanksoftware kann folgendermaßen installiert werden:

- auf demselben Computer wie der Webserver (sowohl MAMP als auch XAMPP installieren MySQL oder MariaDB auf Ihrem Computer)
- auf einem separaten Computer, auf den der Webserver zugreifen kann (so wie sich ein Mailprogramm mit einem Mailserver verbindet)

TABELLEN

In der Datenbank der Beispiel-Website gibt es vier **Tabellen**. Jede Tabelle steht für ein Konzept, das die Anwendung abdeckt:

- **article** steht für jeden einzelnen Artikel und die zugehörigen Daten (z.B. Titel und Erstellungsdatum).
- **category** steht für die Abschnitte der Website, die inhaltlich verwandte Artikel zu Themen zusammenfassen.
- **image** enthält Daten über Bilder, die in Artikeln erscheinen.
- **member** enthält Informationen über die einzelnen Mitglieder der Website.

article								
id	title	summary	content	created	category_id	member_id	image_id	published
1	Systemic Brochure...	This is a...	This brochure...	2021-01-28	1	2	1	1
2	Forecast Handbag...	This is a...	This handbag...	2021-01-28	2	2	2	1
3	Swimming Photos...	This is a...	This is a...	2021-02-02	4	1	3	1

category		
id	name	description
1	Print	Inspiring graphic design
2	Digital	Powerful pixels
3	Illustration	Hand-drawn visual...

Relationale Datenbanken verdanken ihren Namen der Tatsache, dass die Informationen in einer Tabelle mit Daten in anderen Tabellen in Beziehung stehen.

ZEILEN

Jede **Tabellenzeile** (auch Datensatz oder Tupel genannt) enthält eines der Elemente, die die Tabelle darstellt. Jede Zeile in der Tabelle `article` steht für einen Artikel, jede Zeile in der Tabelle `member` für eine Person.

SPALTEN

Jede **Tabellenspalte** (auch Attribut genannt) speichert ein Merkmal der Elemente, die die Tabelle darstellt. In den Spalten der `member`-Tabelle werden beispielsweise der Vorname, der Nachname, die E-Mail-Adresse, das Passwort, das Beitrittsdatum und der Dateiname des Profilbilds gespeichert.

FELDER

Ein **Feld** ist eine einzelne Information aus einer Zeile.

PRIMÄRSCHLÜSSEL

Die erste Spalte in jeder der Tabellen heißt `id`, weil sie zur Identifizierung einer bestimmten Tabellenzeile verwendet werden kann (z.B. zum Auffinden eines einzelnen Artikels, einer Kategorie, eines Mitglieds oder eines Bilds). Damit dies funktioniert, benötigt jede Zeile der Tabelle ihren eigenen eindeutigen Wert in der `id`-Spalte. In diesen Tabellen wird dieser Wert mithilfe der MySQL-Funktion `auto-increment` erzeugt, die zu der in der vorherigen Zeile verwendeten Zahl 1 addiert. Diese Spalte wird auch als **Primärschlüssel** der Tabelle bezeichnet.

image	
id	file
1	systemic-brochure.jpg
2	forecast.jpg
3	swimming-pool.jpg

member						
	id	forename	surname	email	password	joined
ZEILE —	1	Ivy	Stone	ivy@eg.link	c63j-82ve-...	2021-01-26 12:04:23
	2	Luke	Wood	luke@eg.link	saq8-2f2k-...	2021-01-26 12:15:18
	3	Emiko	Ito	emi@eg.link	sk3r-vd92-...	2021-02-12 10:53:47

|
SPALTE

DATENTYPEN IN DATENBANKEN

Sie müssen die einzelnen Spalten einer jeden Tabelle, einen Datentyp und die maximale Zeichenanzahl angeben, die die Felder in dieser Spalte speichern können.

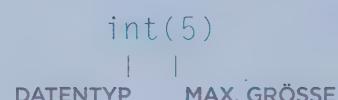
MySQL kennt mehr Datentypen als PHP, aber die Beispieldatenbank nutzt nur die fünf unten aufgeführten.

Hinweis: In MySQL gibt es keinen booleschen Datentyp. Boolesche Werte werden durch den tinyint-Datentyp mit dem Wert 0 für false und 1 für true dargestellt.

DATENTYP	BESCHREIBUNG
int	Ganzzahl
tinyint	Ganzzahl bis 255 (auch für boolesche Werte)
varchar	bis zu 65,535 alphanumerische Zeichen
text	bis zu 65,535 alphanumerische Zeichen
times-tamp	Datum und Uhrzeit

Außer für den Datentyp `text` müssen Sie folgende Höchstwerte angeben:

- maximale Anzahl an Bytes für eine Spalte mit Text
- maximale Stellenanzahl für eine Spalte mit Zahlen



Die Angabe einer maximalen Stellenanzahl für jede Tabellenspalte reduziert die Größe der Datenbank und macht sie schneller. In den folgenden Tabellen sind die Datentypen unter den Spaltennamen aufgeführt, in Klammern gefolgt von der maximalen Anzahl von Ziffern oder Bytes, die die Spalte enthalten kann.

article								
id	title	summary	content	created	category_id	member_id	image_id	published
1	Systemic	Brochure...	Ini ... Brochure	2021-01-26	1	2	1	1
2	Forecast	Handbag...	Init ... 2021-01-28	2021-01-28	3	2	2	1
3	Swimming	Photos...	Imag... 2021-01-30	2021-01-30	4	1	3	1

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual...	1

DUPLIZIERUNG VON DATEN IN EINER DATENBANK VERMEIDEN

Vermeiden Sie die Duplizierung von Daten in einer Datenbank, indem Sie Daten in einer Tabelle über Primär- und Fremdschlüssel mit Daten in einer anderen Tabelle verknüpfen.

In den unteren Tabellen ist die erste Spalte jeweils der **Primärschlüssel**; sie enthält einen Wert, der jede einzelne Tabellenzeile identifiziert. Betrachten Sie zunächst die Tabelle `article` unten links:

- Die Spalte `category_id` zeigt an, in welcher Kategorie sich der Artikel befindet. Ihr Wert stimmt mit einem Primärschlüssel in der Tabelle `category` überein.
- Die Spalte `member_id` zeigt an, wer einen Artikel geschrieben hat. Ihr Wert stimmt mit einem Primärschlüssel in der Tabelle `member` überein.
- Die Spalte `image_id` zeigt an, welches Bild mit dem Artikel angezeigt werden soll. Ihr Wert stimmt mit einem Primärschlüssel in der Tabelle `image` überein.

Die Spalten `category_id`, `member_id` und `image_id` der Tabelle `article` werden als **Fremdschlüssel** bezeichnet. Sie zeigen, dass:

- der erste Artikel sich in der Kategorie `Print` befindet.
- der erste und zweite Artikel von Luke Wood stammen.
- der dritte Artikel das Bild `swimming-pool.jpg` aus der `image`-Tabelle verwendet.

Diese Werte beschreiben die Beziehung zwischen den Daten in verschiedenen Tabellen und ersparen das Duplizieren von Kategorie- und Autorennamen in mehreren Zeilen der Tabelle `article`. Durch den Verzicht auf Doppelungen wird die Datenbank kleiner und schneller, und das Fehlerrisiko sinkt, weil Daten nur an einer Stelle aktualisiert werden müssen, wenn sie sich ändern.

image		
id	file	alt
int(11)	varchar(254)	varchar(1000)
1	systemic-brochure.jpg	Brochure for Systemic Science Festival
2	polite-society-posters.jpg	Posters for Polite Society
3	swimming-pool.jpg	Photography of swimming pool

member						
id	forename	surname	email	password	joined	picture
int(11)	var- char(254)	var- char(254)	var- char(254)	var- char(254)	timestamp	var- char(254)
1	Ivy	Stone	ivy@eg.link	c63j-82ve-...	2021-01-26 12:04:23	ivy.jpg
2	Luke	Wood	luke@eg.link	saq8-2f2k-...	2021-01-26 12:15:18	NULL
3	Emiko	Ito	emi@eg.link	sk3r-vd92-...	2021-02-12 10:53:47	emi.jpg

PHPMYADMIN FÜR DIE ARBEIT MIT MYSQL VERWENDEN

Bei einer datenbankgestützten Website wird die Datenbank normalerweise von den Benutzern aktualisiert, die mit den Seiten der Website interagieren. Manchmal müssen Sie jedoch zusätzliche Aufgaben mit phpMyAdmin durchführen.

PHPMYADMIN EINSETZEN

Beim Betrieb einer datenbankgestützten Website fallen Ihnen bestimmte **Verwaltungsaufgaben** zu, wie etwa:

- Neue Datenbanken erstellen
- Bestehender Datenbanken sichern
- Neue Tabellen und Spalten zu einer Datenbank hinzufügen
- Prüfen, ob die Website die Daten korrekt erfasst oder aktualisiert, nachdem neue Funktionen hinzugefügt wurden
- Benutzerkonten anlegen, die steuern, wer auf den Inhalt einer Datenbank zugreifen beziehungsweise diesen bearbeiten kann

MySQL besitzt keine eigene visuelle Oberfläche, mit der Sie diese Verwaltungsaufgaben durchführen könnten. Es gibt jedoch das kostenlose Open-Source-Tool **phpMyAdmin**, das Sie bei diesen Aufgaben unterstützt.

phpMyAdmin ist in PHP geschrieben und läuft auf Ihrem Webserver; es ist wie eine Website, die Sie zur Verwaltung der Datenbank verwenden können. Auf den folgenden Seiten erfahren Sie, wie Sie damit verschiedene administrative Tätigkeiten durchführen.

SO FINDEN SIE PHPMYADMIN

Nach der Installation von MAMP oder XAMPP steht auch phpMyAdmin auf Ihrem Computer bereit. Geben Sie in Ihrem Browser die URL <http://localhost/phpmyadmin/> ein, um darauf zuzugreifen.

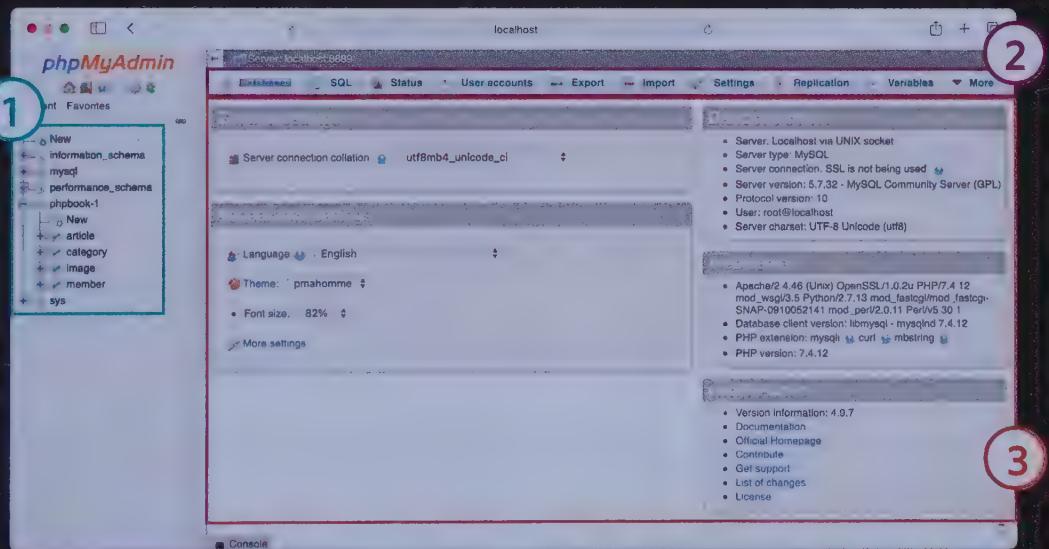
Wenn Sie die URLs um eine Portnummer ergänzen mussten, um die Codebeispiele zum Laufen zu bringen, benötigen Sie für den Zugriff auf phpMyAdmin wahrscheinlich die gleiche Portnummer, z.B.: <http://localhost:8888/phpmyadmin/>.

Hosting-Unternehmen, die MySQL unterstützen, können Ihnen die Durchführung administrativer Aufgaben auf viele verschiedene Arten ermöglichen. Sie müssen also prüfen, welche Lösung Ihr Hosting-Unternehmen Ihnen hier anbietet. Statt eines Zugangs zur Vollversion von phpMyAdmin verwenden sie oft:

- Ihre eigenen Tools zur Erstellung von Datenbanken/ Benutzerkonten
- Eine eingeschränkte Version von phpMyAdmin, um Sicherungskopien zu erstellen und den Inhalt der Datenbank zu überprüfen und zu aktualisieren
- Eigene URLs für den Zugriff auf phpMyAdmin

PHPMYADMIN ZUR VERWALTUNG EINER DATENBANK NUTZEN

Der phpMyAdmin-Bildschirm besteht aus drei Hauptbereichen. Verschiedene Programmversionen können hiervon etwas abweichen, aber die Funktionen und die Positionierung der Elemente auf der Seite sollten gleich sein.



1: DATENBANKEN & TABELLEN

Eine einzelne MySQL-Installation kann mehrere Datenbanken enthalten (so wie ein Webserver mehrere Websites beherbergen kann).

Die Datenbanknamen werden im Menü auf der linken Seite angezeigt. Wenn Sie auf den Namen einer Datenbank klicken, erweitern sich die + -Symbole in der Liste und zeigen die Namen der in dieser Datenbank enthaltenen Tabellen an.

2: REGISTER

Sie stehen für Funktionen, die Sie über die Schnittstelle ausführen können. Wenn Sie:

- phpMyAdmin öffnen, zeigen die Registerkarten Funktionen an, die Sie mit der MySQL-Software ausführen können.
- im linken Menü auf eine Datenbank klicken, ändern sich die Registerkarten in Funktionen, die Sie für diese spezielle Datenbank ausführen können.

3: HAUPTFENSTER

Hier führen Sie administrative Tätigkeiten aus (z.B. wird der Inhalt der Datenbank angezeigt, und Sie können Spalten aktualisieren oder Zeilen hinzufügen).

MySQL erstellt oft eigene Datenbanken wie `information_schema`, `mysql`, `performance_schema` und `sys`. Diese sollten von Anfängern nicht bearbeitet werden.

DIE BEISPIELDATENBANK EINRICHTEN

Um mit den restlichen Beispielen in diesem Buch zu arbeiten, erstellen wir zunächst eine neue Datenbank und importieren einige Daten dorthin.

EINE LEERE DATENBANK ERSTELLEN

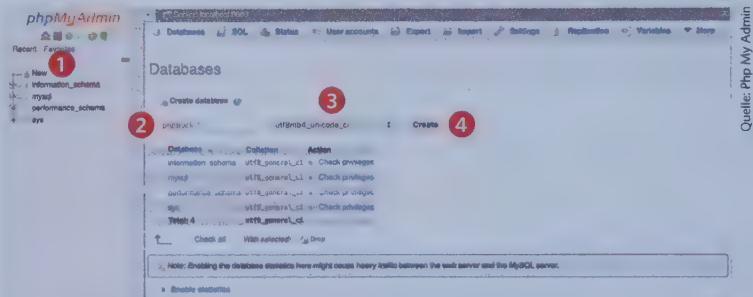
Erstellen Sie zunächst eine leere Datenbank mit phpMyAdmin:

1. Klicken Sie oben in der Datenbankliste auf New.
2. Geben Sie den Datenbanknamen `phpbook-1` ein.
3. Wählen Sie in der Dropdown-Liste Collation den Eintrag `utf8mb4_unicode_ci`, um den Zeichensatz festzulegen.
4. Klicken Sie auf die Schaltfläche Create.

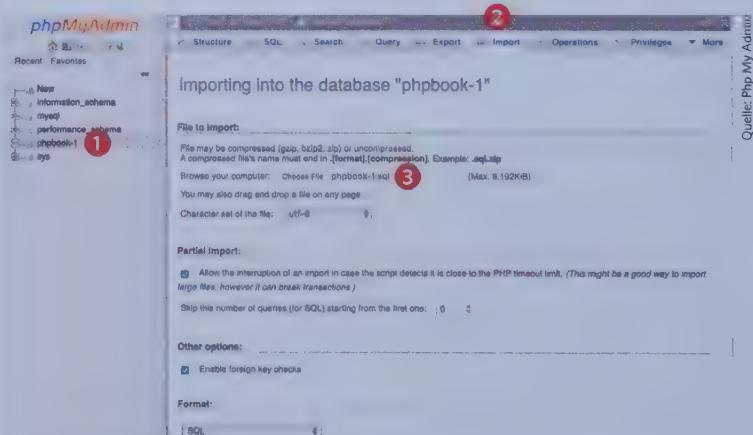
DATEN ZU EINER DATENBANK HINZUFÜGEN

Sobald Sie die Datenbank erstellt haben, können Sie die Daten hinzufügen.

1. Klicken Sie in der Datenbankliste auf den Namen der Datenbank.
2. Klicken Sie auf das Register Import.
3. Klicken Sie unter File to import auf die Schaltfläche Choose file und wählen Sie die Datei `phpbook-1.sql` aus dem Verzeichnis `section_c/intro` folder im Code-Download aus.
4. Klicken Sie auf Go (unten auf der Seite – hier nicht gezeigt), um die Beispieldaten in die Datenbank zu importieren.



Wenn Sie einen Webhosting-Anbieter nutzen, müssen Sie möglicherweise dessen Tools verwenden, um eine leere Datenbank zu erstellen. In diesem Fall können Sie diesen Schritt überspringen.



DIE BEISPIELDATENBANK UNTERSUCHEN

Nachdem Sie die Daten importiert haben, sehen Sie sich die erstellten Tabellen an und untersuchen deren Inhalt und Struktur.

The screenshot shows the phpMyAdmin interface with the following numbered annotations:

- Shows the database structure on the left sidebar.
- Shows the 'article' table selected in the database structure.
- Shows the 'Browse' tab active at the top.
- Shows the first four rows of the 'article' table.

ID	Title	Summary	Content	Created	Category ID	Member ID	Image ID	Published
1	Systems Brochure	Brochure design for a systems brochure for a software developer.	This brochure design is part of a series of designs for a software developer.	2021-01-26 12:21:03	1	2	1	1
2	Drawing Illustration	Illustration for fashion magazine.	This drawing illustration was commissioned by a fashion magazine.	2021-01-28 19:44:00	3	2	2	1
3	Architecture Photography	Photograph of a building's architecture.	This photograph was taken when it was commissioned.	2021-02-09 05:52	4	1	3	1
4	Walking Birds	Article for music video.	The brief for this music video was to create a pa...	2021-02-12 11:05:00	5	3	4	1

DATENBANKINHALTE UNTERSUCHEN

Sie können sich den Inhalt der Datenbank ansehen, indem Sie auf den Namen der Datenbank klicken.

- Klicken Sie in der Datenbankliste auf den Datenbanknamen.
- Wählen Sie die Tabelle **article** aus.
- Klicken Sie auf das Register **Browse**.
- Jede Tabellenzeile steht für einen Artikel.

The screenshot shows the phpMyAdmin interface with the following numbered annotations:

- Shows the database structure on the left sidebar.
- Shows the 'article' table selected in the database structure.
- Shows the 'Structure' tab active at the top.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None			AUTO_INCREMENT Change Drop ↴
2	title	varchar(80)	utf8mb4_unicode_ci		No	None			Change ↴ Drop ↴ ↵
3	summary	varchar(400)	utf8mb4_unicode_ci		No	None			Change ↴ Drop ↴ ↵
4	content	text	utf8mb4_unicode_ci		No				Change ↴ Drop ↴ ↵
5	created	timestamp			No	CURRENT_TIMESTAMP			Change ↴ Drop ↴ ↵
6	category_id	int(11)			No	None			Change ↴ Drop ↴ ↵
7	member_id	int(11)			No	None			Change ↴ Drop ↴ ↵
8	image_id	int(11)			Yes	NULL			Change ↴ Drop ↴ ↵
9	published	tinyint(1)			No	0			Change ↴ Drop ↴ ↵

TABELLENSTRUKTUR

Für jede Tabelle können Sie die Spaltennamen, Datentypen und Feldgrößen sehen.

- Klicken Sie auf den Datenbanknamen.
- Wählen Sie die Tabelle **article** aus.
- Klicken Sie auf das Register **Structure**.

Jede Spalte wird in der angezeigten Tabelle als eine Zeile aufgeführt.

Wie Sie Tabellen und Spalten manuell hinzufügen können, erfahren Sie unter:

<http://notes.re/mysql/create-manually>

Sie können den Namen jeder Spalte sehen, gefolgt vom Datentyp (mit der maximalen Größe in Klammern). Collation enthält die Zeichenkodierung.

Null gibt an, ob ein Wert null sein darf. Die Spalte Default gibt einen Standardwert an, den die Spalte verwenden soll, wenn kein Wert angegeben wurde.

BENUTZERKONTEN FÜR DATENBANKEN ANLEGEN

Mit der MySQL-Software können Sie verschiedene Benutzerkonten anlegen. Jedes dieser Konten hat einen Benutzernamen und ein Passwort zur Anmeldung an der Datenbank. Sie können festlegen, auf welche Daten ein Benutzerkonto zugreifen und welche es aktualisieren darf.

Für jedes MySQL-Benutzerkonto können Sie angeben:

- auf welche Datenbanken der Benutzer zugreifen kann
- welche Tabellen er aufrufen oder aktualisieren kann
- andere Aufgaben, die der Benutzer ausführen kann

Bei der Installation von MySQL wird ein sogenanntes **Root-Konto** eingerichtet, ein Konto mit übergeordneten Rechten, das Benutzerkonten und Datenbanken erstellen und löschen kann.

Aus Sicherheitsgründen sollten Sie das Root-Konto nicht in Ihrem PHP-Code verwenden. Erstellen Sie stattdessen ein Benutzerkonto mit eingeschränkten Rechten:

- Zugriff auf die Datenbank, die für die betreffende Website verwendet wird (nicht auf alle Datenbanken, die auf demselben Server gehostet werden)
- Ausführen von Aufgaben, die von der Anwendung benötigt werden – erlauben Sie nicht die Ausführung mächtiger Aufgaben (wie das Erstellen oder Löschen von Tabellen), wenn dies für die Website nicht erforderlich ist

Rechts sehen Sie, wie Sie mit dem bei der Installation des Servers eingerichteten Root-Konto Benutzerkonten in phpMyAdmin anzeigen und erstellen können.

Wenn Sie einen Webhosting-Anbieter verwenden und diese Optionen nicht sehen können, sollten Sie die Hilfdateien des Anbieters konsultieren, da hier jeder Anbieter anders arbeitet und möglicherweise:

- einen Benutzernamen und ein Passwort für Sie erstellt hat
- eigene Tools zum Anlegen und Aktualisieren von Benutzern verwendet

1. Wählen Sie im linken Bedienfeld den Namen der Datenbank aus, für die Sie einen Benutzer anlegen möchten.
2. Klicken Sie auf die Registerkarte **Privileges**. Daraufhin erscheint eine Tabelle mit den Benutzern, die Zugriff auf diese Datenbank haben.
3. Klicken Sie auf **Add user** oder **Add user account**.
4. Geben Sie einen Benutzernamen an.
5. Geben Sie ein Passwort ein oder verwenden Sie die Option **Generate**.
6. Deaktivieren Sie unter **Database for user account** die Option **Grant all privileges on database phpbook -1**, falls diese aktiviert ist.
7. Die Optionen für **Global privileges** steuern, was der Benutzer mit der Datenbank insgesamt tun darf. Die Beispiel-Website benötigt nur die vier Optionen, die in der Spalte **Data** ausgewählt wurden: **Select, Insert, Update und Delete**; die anderen Funktionen werden nicht verwendet und sind daher nicht aktiviert.
8. Speichern Sie die Benutzerangaben mit der Schaltfläche **Go** am unteren Seitenrand (in den Screenshots nicht dargestellt).

Sie können die Tabellen im linken Bereich auch einzeln auswählen und die Benutzerrechte dafür festlegen.

1 Recent

2 Structure SQL Search Query Export Import Operations Privileges Routines More

3 Add user account

Users having access to "phpbook-1"

User name	Host name	Type	Privileges	Grant	Action
root	localhost	global	ALL PRIVILEGES	Yes	

Check all With selected:

New

Add user account **3**

Quelle: Php My Admin

4 User name: testuser

5 Host name: Any host

6 Database for user account

Create database with same name and grant all privileges.
Grant all privileges on wildcard name (username_{wildcard}).
Grant all privileges on database phpbook-1.

7 Global privileges Check all

Note: MySQL privilege names are expressed in English

Data	Structure	Administration	Resource limits
<input checked="" type="checkbox"/> SELECT <input checked="" type="checkbox"/> INSERT <input checked="" type="checkbox"/> UPDATE <input checked="" type="checkbox"/> DELETE <input type="checkbox"/> FILE	CREATE ALTER INDEX DROP CREATE TEMPORARY TABLES SHOW VIEW CREATE ROUTINE ALTER ROUTINE EXECUTE CREATE VIEW EVENT TRIGGER	GRANT SUPER PROCESS RELOAD SHUTDOWN SHOW DATABASES LOCK TABLES REFERENCES REPLICATION CLIENT REPLICATION SLAVE CREATE USER	Note: Setting these options to 0 (zero) removes the limit. MAX QUERIES PER HOUR: 0 <input type="button" value="+"/> <input type="button" value="-"/> MAX UPDATES PER HOUR: 0 <input type="button" value="+"/> <input type="button" value="-"/> MAX CONNECTIONS PER HOUR: 0 <input type="button" value="+"/> <input type="button" value="-"/> MAX USER CONNECTIONS: 0 <input type="button" value="+"/> <input type="button" value="-"/>

SSL

REQUIRE NONE
REQUIRE SSL
REQUIRE X509
SPECIFIED
REQUIRE CIPHER
REQUIRE ISSUER

Console

Quelle: Php My Admin

KAPITEL IN TEIL C

DATENBANKGESTÜTZTE WEBSITES

WICHTIGER HINWEIS:

Für die restlichen Buchkapitel müssen Sie den Code von <http://phpandmysql.com/code/> herunterladen und lokal ausführen. Es ist auch hilfreich, beim Lesen dieser Kapitel den Beispiel-Code geöffnet zu haben.

11

STRUCTURED QUERY LANGUAGE

SQL ist eine Sprache, mit der Sie festlegen können, welche Daten Sie aus der Datenbank abrufen und welche Daten darin aktualisiert werden sollen. Dieses Kapitel vermittelt die Funktionsweise von SQL, indem es Sie SQL-Befehle in die Benutzeroberfläche von phpMyAdmin eingeben lässt.

12

DATEN AUS DER DATENBANK ABRUFEN

Nachdem Sie SQL kennengelernt haben, erfahren Sie, wie PDO eine SQL-Anweisung an die Datenbank senden und wie PHP auf die zurückgelieferten Daten zugreifen kann. Die von der Datenbank zurückgegebenen Daten können Ihrem PHP-Code entweder als Array oder als Objekt zur Verfügung gestellt werden.

13

DATEN IN DER DATENBANK AKTUALISIEREN

In diesem Kapitel lernen Sie, wie Sie Daten von einem Website-Besucher empfangen, wie Sie diese validieren und schließlich zur Aktualisierung der Datenbank verwenden können. Sie erfahren zudem, wie Sie Problemen begegnen, die eine Aktualisierung der Datenbank verhindern.

11

STRUCTURED QUERY LANGUAGE

Structured Query Language (SQL) ist eine Sprache, die für die Kommunikation mit Datenbanken verwendet wird. Mit ihr können Sie Daten abfragen, neue Daten hinzufügen, bestehende Daten bearbeiten und Daten löschen.

In diesem Kapitel lernen Sie, wie Sie mit SQL die folgenden Aufgaben durchführen können:

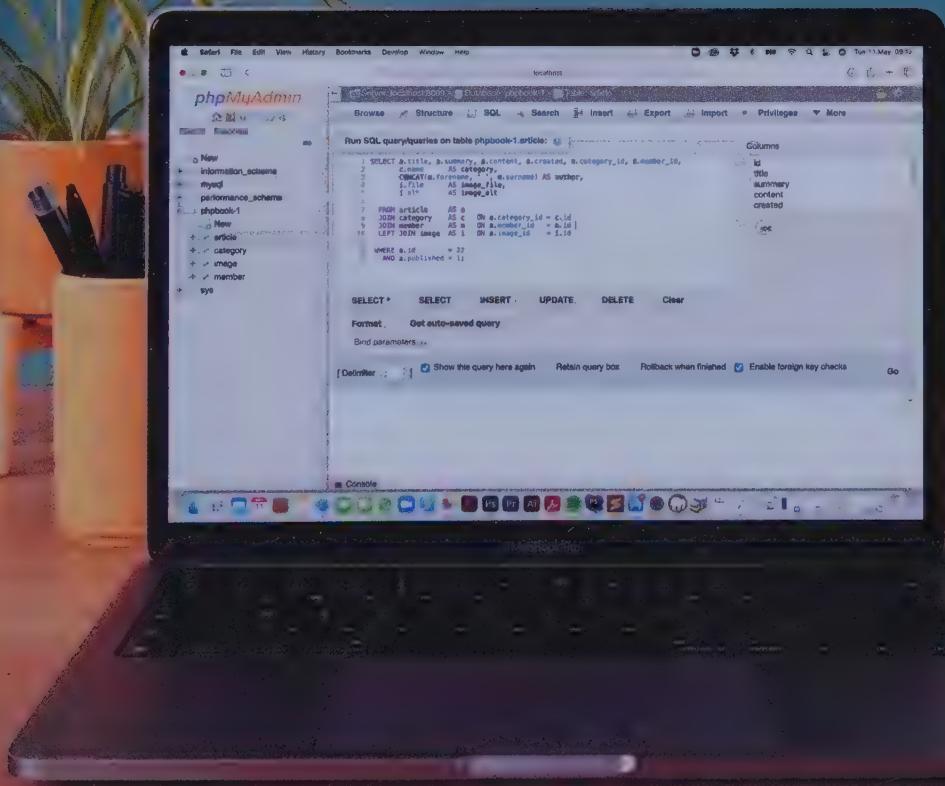
- Daten aus einer Datenbank **wählen**
- Neue Zeilen in einer Datenbanktabelle **erstellen**
- bereits in einer Datenbank gespeicherten Daten **aktualisieren**
- Zeilen aus Datenbanktabellen **löschen**

Eine Anweisung zum Abrufen oder Ändern von in der Datenbank gespeicherten Daten wird als **SQL-Anweisung** bezeichnet.

Eine SQL-Anweisung, die nur nach Informationen fragt, kann auch als **SQL-Abfrage** bezeichnet werden, da Sie die Datenbank nach Daten *fragen*. Sie werden lernen, wie man SQL-Abfragen schreibt, bevor Sie lernen, wie man Daten in einer Datenbank erstellt, aktualisiert oder löscht.

Um die SQL-Sprache zu erlernen, werden Sie phpMyAdmin verwenden. Nachdem Sie in diesem Kapitel gelernt haben, wie SQL funktioniert, werden Sie in den nächsten beiden Kapiteln erfahren, wie eine PHP-Seite mit PDO (PHP Data Objects) SQL-Anweisungen ausführt, die Daten in der Datenbank abrufen oder aktualisieren.

Einige Beispiele in diesem Kapitel aktualisieren die in der Datenbank gespeicherten Daten, die die Grundlage für die Website des Hauptbeispiels in diesem Buch bildet, sodass sie einmal in der Reihenfolge ausgeführt werden sollten, in der sie in diesem Kapitel erscheinen. Sonst funktionieren die späteren Beispiele möglicherweise nicht. In diesem Fall oder wenn Sie die Beispiele erneut ausführen möchten, löschen Sie die Datenbank und richten sie erneut ein, indem Sie die Anweisungen in der Einleitung zu diesem Abschnitt befolgen.



DATEN AUS EINER DATENBANK ABRUFEN

Um Daten aus der Datenbank abzufragen, verwenden Sie den SQL-Befehl SELECT und geben dann an, welche Daten Sie zurückgeben möchten. Die Datenbank erstellt daraufhin eine **Ergebnismenge** mit den gewünschten Daten.

Der SELECT-Befehl gibt an, dass Sie Daten aus der Datenbank abrufen wollen. Ihm folgen die Namen der Spalten, die die gewünschten Daten enthalten. Jeder Spaltenname sollte durch ein Komma getrennt werden.

Nach der FROM-Klausel folgt der Name der Tabelle, aus der Sie die Daten abrufen wollen. Eine SQL-Anweisung sollte mit einem Semikolon abgeschlossen werden (obwohl viele Entwickler dies weglassen und es normalerweise trotzdem funktioniert).

AUSZUWÄLLENDE SPALTEN → *SELECT column1, column2
TABELLENSPALTEN BEFINDEN SICH IN*

Die folgende SQL-Anweisung fragt nach den Daten in den Spalten forename und surname der Tabelle member. Sie gibt die Daten aus jeder Zeile der Tabelle zurück.

Sie können die Anweisung wörtlich lesen; sie sagt Folgendes aus:

- Wähle (SELECT) die Spalten forename und surname
- aus (FROM) der Tabelle member

SPALTE 1 SPALTE 2

 | |

```
SELECT forename, surname
  FROM member;
  |
  TABELLE
```

SQL-Befehle können in Groß- oder Kleinbuchstaben geschrieben werden. In diesem Buch werden sie in Großbuchstaben geschrieben, um sie von Tabellen- und Spaltennamen zu unterscheiden. Bei Tabellen- und Spaltennamen muss die gleiche Groß- und Kleinschreibung wie in der Datenbank verwendet werden.

Wenn eine SQL-Abfrage **ausgeführt** wird, ruft die Datenbank die angeforderten Daten ab und legt sie in einer Ergebnismenge ab. Die Spalten werden der Ergebnismenge in derselben Reihenfolge hinzugefügt, in der sie in der Abfrage genannt werden. Um die Reihenfolge zu steuern, in der Zeilen zur Ergebnismenge hinzugefügt werden, verwenden Sie die ORDER BY-Klausel (siehe S. 406).

Ergebnismenge	
forename	surname
Ivy	Stone
Luke	Wood
Emiko	Ito

Die rechte Seite zeigt, wie man die SQL-Abfrage in phpMyAdmin eingibt und wie die Ergebnismenge, die die Abfrage erzeugt, angezeigt wird.

1. Öffnen Sie phpMyAdmin und wählen Sie die Datenbank phpbook-1. Falls Sie sie noch nicht erstellt haben, siehe S. 392.
2. Wählen Sie die Registerkarte SQL.

The screenshot shows the phpMyAdmin interface with the SQL tab selected. A red circle labeled '1' is on the 'phpbook-1' database node in the left sidebar. A red circle labeled '2' is at the top center of the window. A red circle labeled '3' is over the SQL query input field containing:

```
1 SELECT forename, surname
2 FROM member;
```

Below the query are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Show this query here again (checked), Retain query box, Delimiter [;], Rollback when finished (checked), Enable foreign key checks, Go (button), and a red circle labeled '4'.

5. Sobald Sie auf Go klicken, wird die SQL-Abfrage ausgeführt. MySQL gibt dann die Ergebnismenge an phpMyAdmin zurück, und phpMyAdmin zeigt die Ergebnismenge in einer Tabelle an.

Im weiteren Verlauf dieses Kapitels werden keine Screenshots von phpMyAdmin gezeigt, sondern SQL-Abfragen und deren Ergebnisse wie auf der linken Seite.

The screenshot shows the phpMyAdmin interface with the Browse tab selected. A red circle labeled '5' is over the table results. The table has columns 'forename' and 'surname'. The data is:

forename	surname
Ivy	Stone
Luke	Wood
Emiko	Ito

Below the table are buttons: Edit, Copy, Delete, Check all, With selected: Edit, Copy, Delete, Export, and a red circle labeled '4'.

BESTIMMTE TABELLENZEILEN ZURÜCKGEBEN

Um Daten aus bestimmten (und nicht aus allen) Tabellenzeilen abzurufen, fügen Sie die WHERE-Klausel gefolgt von einer **Suchbedingung** hinzu.

Nachdem Sie festgelegt haben, welche Datenspalten aus einer Tabelle abgerufen werden sollen, können Sie eine Suchbedingung hinzufügen, um zu steuern, welche Zeilen dieser Tabelle der Ergebnismenge hinzugefügt werden sollen. In der Suchbedingung geben Sie eine Spalte der Tabelle an und legen fest, ob ihr Wert gleich =, größer als > oder kleiner als < ein von Ihnen festgelegter Wert sein soll.

Wenn die Datenbank die Zeilen der Tabelle abarbeitet und die Bedingung erfüllt ist, wird der Ergebnismenge eine neue Zeile hinzugefügt, und die Daten aus den nach dem SELECT-Befehl genannten Spalten werden in die Ergebnismenge kopiert.

AUSZUWÄHLENDE SPALTEN →

SPALTEN BEFINDEN SICH IN →

DER ERGEBNISMENGE HINZUZUFÜGENDE ZEILEN →

Wenn der Wert, den Sie in der Bedingung angeben, Text ist, setzen Sie den Text in einfache Anführungszeichen.

Wenn der Wert, den Sie in der Bedingung angeben, eine Zahl ist, setzen Sie die Zahl nicht in Anführungszeichen.

MySQL verfügt über keinen booleschen Datentyp, aber Sie können einen tinyint-Datentyp verwenden, um einen booleschen Wert mit 1 für true und 0 für false darzustellen. Da es sich um Zahlnwerte handelt, sollten sie nicht in Anführungszeichen gesetzt werden.

```
SELECT column(s)  
      FROM table  
 WHERE column = value;  
       |  
       OPERATOR
```

Sie können mehrere Suchbedingungen mit den drei logischen Operatoren aus der Tabelle rechts kombinieren. Sie funktionieren wie die logischen Operatoren von PHP (siehe S. 56–57). Jede Suchbedingung wird in Klammern gesetzt, um sicherzustellen, dass sie für sich alleine ausgeführt wird.

AUSZUWÄHLENDE SPALTEN → SELECT column(s)

SPALTEN BEFINDEN SICH IN → FROM table

DER ERGEBNISMENGE → WHERE (column < value) AND (column > value);
HINZUZUFÜGENDE ZEILEN

OPERATOR BESCHREIBUNG

AND Alle Bedingungen müssen wahr sein.

OR Jede einzelne Bedingung kann wahr sein.

NOT Kehrt eine Bedingung um; prüft, ob sie nicht wahr ist.

BEDINGUNG LOGISCHER OPERATOR BEDINGUNG

VERGLEICHSOPERATOREN IN SQL VERWENDEN

SQL

section_c/c11/comparison-operator-1.sql

```
SELECT email
  FROM member
 WHERE forename = 'Ivy';
```

Ergebnismenge

email
ivy@eg.link

SQL

section_c/c11/comparison-operator-2.sql

```
SELECT email
  FROM member
 WHERE id < 3;
```

Ergebnismenge

email
ivy@eg.link
luke@eg.link

SQL

section_c/c11/logical-operator.sql

```
SELECT email
  FROM member
 WHERE (email > 'E') AND (email < 'L');
```

Ergebnismenge

email
emi@eg.link
ivy@eg.link

In diesem Beispiel werden die E-Mail-Adressen aller Mitglieder ausgewählt, die in der Spalte forename der Tabelle member den Wert Ivy haben. Geben Sie den SQL-Code in phpMyAdmin ein, wie auf S. 401 gezeigt.

Probieren Sie es: Finden Sie die E-Mail-Adressen aller Mitglieder, die Luke heißen.

In diesem Beispiel werden die E-Mail-Adressen von Mitgliedern gesucht, die in der Spalte id der Tabelle member einen Wert von weniger als 3 haben.

Probieren Sie es: Finden Sie die E-Mail-Adressen der Mitglieder, deren id kleiner oder gleich 3 ist.

In diesem Beispiel werden mit dem Operator AND Ergebnisse gefunden, bei denen die E-Mail-Adresse eines Mitglieds zwei Bedingungen erfüllt – sie beginnt mit:

1. einem Buchstabe größer als E
2. einem Buchstaben kleiner als L

Probieren Sie es: Suchen Sie die E-Mail-Adressen aller Mitglieder, deren E-Mail-Adressen mit den Buchstaben G–L beginnen.

ERGEBNISSE MIT LIKE & WILDCARDS SUCHEN

Der LIKE-Operator kann in einer Suchbedingung verwendet werden, um Datenzeilen zu finden, in denen der Wert in einer bestimmten Spalte mit bestimmten Zeichen beginnt, endet oder diese enthält.

Der LIKE-Operator findet Datenzeilen, in denen eine Spalte Zeichen enthält, die einem von Ihnen angegebenen **Muster** entsprechen. Sie können das Muster zum Beispiel verwenden, um Zeilen zu finden, in denen der Wert in einer bestimmten Spalte

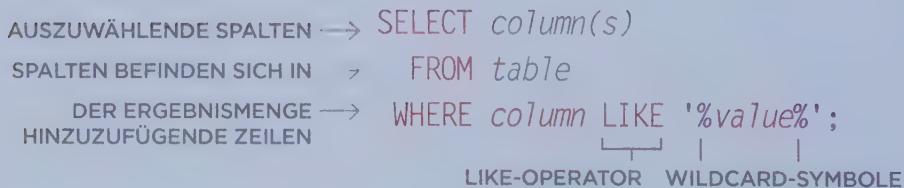
- mit einem bestimmten Buchstaben beginnt,
- mit einer bestimmten Zahl endet oder
- ein bestimmtes Wort oder eine bestimmte Zeichenfolge enthält (dies wird häufig zum Erstellen von Suchfunktionen verwendet).

Ein Muster kann auch **Wildcard-Symbole** enthalten, die anzugeben, wo andere Zeichen stehen könnten (wie in der Tabelle auf der rechten Seite gezeigt):

- % steht für null oder mehr Zeichen
- _ steht für ein einzelnes Zeichen

WERT	FINDET SPALTEN, DEREN WERT ...
To%	mit To beginnt
%day	mit day endet
%to%	an beliebiger Stelle to enthält
h_11	ein einzelnes Zeichen an der Stelle des Unterstrichs enthält (z.B. hall, he11, hill1, hull1)
%h_11%	enthält h, dann ein beliebiges Zeichen, dann 11 (z.B. hall, hell1, chill, hillly, shellac, chilled, hallmark, hullabaloo)
1%	beginnt mit 1
%!	endet mit !

Der Wert ist unabhängig von der Groß- und Klein-schreibung, d. h., die Suche nach dem Namen Ivy würde auch die Werte IVY und ivy finden.



WERTE SUCHEN

SQL

section_c/c11/like-1.sql

```
SELECT email  
      FROM member  
     WHERE forename LIKE 'I%';
```

Ergebnismenge

email
ivy@eg.link

SQL

section_c/c11/like-2.sql

```
SELECT email  
      FROM member  
     WHERE forename LIKE 'E_I%';
```

Ergebnismenge

email
emi@eg.link

SQL

section_c/c11/like-3.sql

```
SELECT email  
      FROM member  
     WHERE forename LIKE 'Luke';
```

Ergebnismenge

email
luke@eg.link

In diesem Beispiel wird nach den E-Mail-Adressen aller Mitglieder gesucht, deren Vorname mit dem Buchstaben I beginnt (Groß- oder Kleinschreibung).

Probieren Sie es: Finden Sie Mitglieder, deren Name mit dem Buchstaben E beginnt.

Dieses Beispiel ruft die E-Mails aller Mitglieder ab, deren Vornamen

- mit dem Buchstaben E beginnen,
- worauf ein weiterer Buchstabe folgt,
- dann der Buchstabe I,
- dann ein beliebiges weiteres Zeichen.

(Es würden die Namen Eli, Elias, Elijah, Elisha, Emi, Emiko, Emil, Emilio, Emily, Eoin und Eric zurückgegeben.)

Probieren Sie es: Suchen Sie nach Mitgliedern, deren Name L_K% entspricht.

In diesem Beispiel wird nach der E-Mail-Adresse einer Person gesucht, deren Name Lukas ist. Da es keine Platzhalterzeichen gibt, werden nur exakte Übereinstimmungen zurückgegeben.

Probieren Sie es: Finden Sie Mitglieder mit dem Namen Ivy.

ZEILENREIHENFOLGE IN EINER ERGEBNISMENGE STEUERN

Um die Reihenfolge zu steuern, in der Zeilen zu einer Ergebnismenge hinzugefügt werden, verwenden Sie die ORDER BY-Klausel, gefolgt vom Namen einer Spalte, deren Werte die Ergebnisse ordnen sollen, und dann entweder ASC für aufsteigend oder DESC für absteigend.

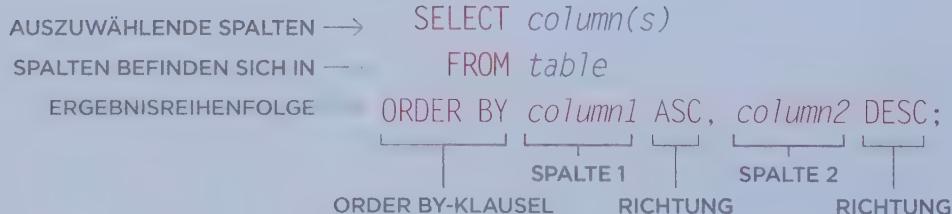
ORDER BY kann am Ende einer Abfrage hinzugefügt werden, um die Reihenfolge zu steuern, in der die Zeilen zur Ergebnismenge hinzugefügt werden. Mit den Werten in der angegebenen Spalte wird die Reihenfolge der Ergebnisse gesteuert.

Darauf sollte eines von zwei Schlüsselwörtern folgen:
ASC für aufsteigend oder DESC für absteigend. (Wenn Sie keines der beiden Schlüsselwörter angeben, wird die Sortierung in aufsteigender Reihenfolge durchgeführt, aber Ihr SQL ist leichter zu lesen, wenn Sie ASC oder DESC angeben.)



Sie können die Reihenfolge, in der Zeilen zur Ergebnismenge hinzugefügt werden, anhand der Werte aus mehreren Spalten sortieren. Jeder Spaltenname wird durch ein Komma getrennt. Wenn die erste Spalte, die zum Sortieren von Werten verwendet wird, identische Werte enthält, verweist sie auf die zweite Spalte in der Liste.

Wenn Sie beispielsweise die Mitglieder nach ihrem Namen sortieren und mehrere Mitglieder denselben Vornamen haben (der in der Spalte `forename` gespeichert ist), können Sie sie nach ihrem Nachnamen sortieren (der in der Spalte `surname` gespeichert ist).



ERGEBNISSE SORTIEREN

SQL

section_c/c11/order-by-1.sql

```
SELECT email  
      FROM member  
 ORDER BY email DESC;
```

In diesem Beispiel werden alle E-Mail-Adressen abgerufen und in absteigender Reihenfolge sortiert.

Probieren Sie es: Kehren Sie die Reihenfolge der Ergebnisse um, indem Sie DESC in ASC ändern.

Ergebnismenge

email
luke@eg.link
ivy@eg.link
emi@eg.link

SQL

section_c/c11/order-by-2.sql

```
SELECT title, category_id  
      FROM article  
 ORDER BY category_id ASC, title ASC;
```

Dieses Beispiel ruft Werte aus den Spalten title und category_id der Tabelle article ab und ordnet sie zuerst in aufsteigender Reihenfolge nach der category_id und dann in alphabetischer Reihenfolge nach dem Titel.

Die gesamte Ergebnismenge enthält mehr Zeilen, als links gezeigt werden (eine für jeden Artikel), aber der Platz reicht nicht aus, um sie hier alle zu zeigen.

Probieren Sie es: Wählen Sie die Spalten title und member_id aus der Tabelle article aus und ordnen Sie die Ergebnisse zuerst nach der member_id und dann nach dem Titel in alphabetischer Reihenfolge.

Ergebnismenge (zeigt die ersten 10 von 24 Zeilen)

title	category_id
Chimney Business Cards	1
Milk Beach Album Cover	1
Polite Society Posters	1
Systemic Brochure	1
The Ice Palace	1
Travel Guide	1
Chimney Press Website	2
Floral Website	2
Milk Beach Website	2
Polite Society Website	2

ERGEBNISSE ZÄHLEN UND GRUPPIEREN

Wenn die SQL-Funktion COUNT() nach dem SELECT-Befehl verwendet wird, wird die Gesamtzahl der Zeilen, die der Abfrage entsprechen, zur Ergebnismenge hinzugefügt. Durch Gruppieren der Ergebnisse können Sie zählen, wie viele Zeilen denselben Wert enthalten.

Um die Anzahl der Zeilen in einer Tabelle zu zählen, rufen Sie COUNT() nach dem SELECT-Befehl auf und geben die Tabelle an. Verwenden Sie als Argument für die Funktion COUNT() ein Sternchen (ein sogenanntes Wildcard).

Wenn Sie der Abfrage eine Suchbedingung hinzufügen, gibt die Funktion COUNT() die Anzahl der Zeilen zurück, die der Abfrage mit der Suchbedingung entsprechen.

Wenn Sie einen Spaltennamen als Argument für die Funktion COUNT() angeben, zählt sie die Anzahl der Zeilen, in denen der Wert in der angegebenen Spalte nicht null ist.

Die GROUP BY-Klausel kann mit der SQL-Funktion COUNT() verwendet werden, um festzustellen, wie viele Zeilen denselben Wert in einer Spalte haben. So kann z.B. gezählt werden, wie viele Artikel ein Mitglied geschrieben hat oder wie viele Artikel sich in einer Kategorie befinden. In der SELECT-Anweisung ...

1. wählen Sie einen Spaltennamen, der identische Werte enthalten kann (z.B. member_id oder category_id).
2. verwenden Sie COUNT(*), um die Anzahl der Zeilen zu zählen.
3. geben Sie die Tabelle an, in der sich die Spalte befindet.
4. verwenden Sie die GROUP BY-Klausel, gefolgt vom Namen der Spalte, die identische Werte enthalten kann, sodass die Anzahl der Zeilen mit demselben Wert in dieser Spalte gruppiert und gezählt wird.

FUNKTION
SELECT COUNT(*)
FROM table;

SELECT COUNT(*)
FROM table
WHERE column LIKE '%value%';

SELECT COUNT(column)
FROM table;

SPALTE HAT EVTL.
DENSELBN WERT
SELECT column, COUNT(*)
FROM table
GROUP BY column;

SPALTE HAT EVTL.
DENSELBN WERT

ANZAHL ÜBEREINSTIMMENDER ERGEBNISSE ZÄHLEN

SQL

section_c/c11/count-1.sql

```
SELECT COUNT(picture)  
FROM member;
```

Ergebnismenge

COUNT(picture)
2

SQL

section_c/c11/count-2.sql

```
SELECT COUNT(*)  
FROM article  
WHERE title LIKE '%design%' OR content LIKE '%design%';
```

Ergebnismenge

COUNT(*)
9

SQL

section_c/c11/count-3.sql

```
SELECT member_id, COUNT(*)  
FROM article  
GROUP BY member_id;
```

Ergebnismenge

member_id	COUNT(*)
1	10
2	8
3	6

In diesem Beispiel wird mit der Funktion COUNT() von SQL die Anzahl der Mitglieder ermittelt, die ein Profilbild angegeben haben. Wenn der Wert in der Bildspalte NULL ist, werden sie nicht gezählt.

Probieren Sie es: Zählen Sie die Anzahl der Mitglieder mit einer E-Mail-Adresse.

In diesem Beispiel wird mit der SQL-Funktion COUNT() die Anzahl der Artikel ermittelt, in deren title- oder content-Spalte der Begriff Design vorkommt.

Probieren Sie es: Finden Sie die Anzahl der Artikel, die den Begriff photo enthalten.

Diese Abfrage zählt die Anzahl der Artikel der einzelnen Mitglieder.

Die SELECT-Anweisung ruft die Spalte member_id ab, und die Funktion COUNT() zählt die Anzahl der übereinstimmenden Zeilen.

Die FROM-Klausel zeigt an, dass in der Tabelle article gesucht werden soll. Die GROUP BY-Klausel gruppiert die Werte in der Spalte member_id, sodass Sie die ID des Mitglieds und die Anzahl der von ihm verfassten Artikel sehen.

Probieren Sie es: Berechnen Sie die Anzahl der Artikel in jeder Kategorie.

ERGEBNISSE EINSCHRÄNKEN UND ÜBERSPRINGEN

`LIMIT` schränkt die Anzahl der Ergebnisse ein, die einer Ergebnismenge hinzugefügt werden. `OFFSET` weist die Datenbank an, eine bestimmte Anzahl von Datensätzen zu überspringen und die nachfolgenden zur Ergebnismenge hinzuzufügen.

Um die Gesamtzahl der Zeilen zu begrenzen, die der Ergebnismenge hinzugefügt werden, verwenden Sie die `LIMIT`-Klausel.

Das folgende Beispiel würde nur die ersten fünf Ergebnisse, die mit der Abfrage übereinstimmen, zur Ergebnismenge hinzufügen.

Die **OFFSET**-Klausel kann nach der **LIMIT**-Klausel verwendet werden, um die ersten Übereinstimmungen zu überspringen, die sonst der Ergebnismenge hinzugefügt worden wären.

Das folgende Beispiel überspringt die ersten sechs Ergebnisse, die mit der Abfrage übereinstimmen, und fügt die folgenden drei Ergebnisse zur Ergebnismenge hinzu.

The diagram illustrates the structure of a SELECT query. It starts with 'AUSZUWÄLLENDE SPALTEN →' followed by 'SELECT column(s)'. Below it, 'SPALTEN BEFINDEN SICH IN →' is followed by 'FROM table'. A horizontal line connects these two parts. Below the line, 'ZEILENANZAHL BEGRENZEN UND ÜBERSPRINGEN →' is followed by 'LIMIT 3 OFFSET 6;'. An annotation shows a bracket under 'OFFSET 6;' with labels: 'OFFSET-KLAUSEL' on the left and 'ZU ÜBERSPRINGENDE ERGEBNISSE' on the right, with a vertical line pointing down to the '6;'.

LIMIT und OFFSET werden häufig verwendet, wenn eine Abfrage sehr viele Ergebnisse liefert. Die Ergebnisse werden mit einer Technik namens Paginierung auf verschiedene Seiten aufgeteilt. Ein bekanntes Beispiel sind die Suchergebnisse von Google.

Nach der ersten Ergebnisseite gibt es Links zu weiteren Seiten, die weitere Ergebnisse für dieselbe Abfrage anzeigen. Wie Sie mit diesen Befehlen eine Paginierung hinzufügen können, erfahren Sie im nächsten Kapitel.

ANZAHL ÜBEREINSTIMMENDER ERGEBNISSE EINSCHRÄNKEN

SQL

```
SELECT title  
      FROM article  
     ORDER BY id  
    LIMIT 1;
```

section_c/c11/limit.sql

In diesem Beispiel werden die Artikeltitel abgefragt, sortiert nach dem Wert in der Spalte `id`. Es verwendet die `LIMIT`-Klausel, um nur die erste Übereinstimmung in die Ergebnismenge aufzunehmen.

Probieren Sie es: Holen Sie sich die ersten fünf Artikel aus der Kategorie `Print`.

Ergebnismenge

title
Systemic Brochure

SQL

```
SELECT title  
      FROM article  
     ORDER BY id  
    LIMIT 3 OFFSET 9;
```

section_c/c11/offset.sql

In diesem Beispiel werden die Artikeltitel abgefragt, sortiert nach dem Wert in der Spalte `id`. Es überspringt mit der `OFFSET`-Klausel die ersten neun Ergebnisse, die der Abfrage entsprechen, und fügt dann mit der `LIMIT`-Klausel die nächsten drei Treffer zur Ergebnismenge hinzu.

Probieren Sie es: Überspringen Sie die ersten sechs Ergebnisse und geben Sie die nächsten sechs zurück.

Ergebnismenge

title
Polite Society Mural
Stargazer Website and App
The Ice Palace

DATEN MIT JOINS AUS ZWEI TABELLEN ABRUFEN

Mit einem **Join** können Sie Daten aus mehr als einer Tabelle abfragen. Die Daten aus beiden Tabellen werden zu einer einzigen Zeile in der Ergebnismenge hinzugefügt.

Wenn Sie eine Datenbank entwerfen, sollten Sie für jedes Konzept der Website eine Tabelle erstellen und Datenduplikate in mehreren Tabellen vermeiden.

Auf der Beispiel-Website befinden sich die Daten zu Artikeln, Kategorien, Mitgliedern und Bildern in verschiedenen Tabellen. Die erste Spalte in diesen Tabellen enthält einen Wert, der zur Identifizierung jeder Tabellenzeile verwendet wird. Der Wert in der `id`-Spalte der Tabelle `category` kann zum Beispiel jede Kategorie identifizieren. Dieser Wert wird als **Primärschlüssel** bezeichnet.

In der Tabelle `article` muss gespeichert werden, welcher Kategorie die einzelnen Artikel angehören. Anstatt die Kategorienamen für jeden Artikel zu duplizieren, gibt es eine Spalte `category_id`. Der Wert in dieser Spalte wird als **Fremdschlüssel** bezeichnet und entspricht dem Primärschlüssel der Kategorie, in der er sich befindet.

Die Primär- und Fremdschlüssel beschreiben, wie sich Daten in einer Tabellenzeile auf Daten in einer anderen Tabellenzeile **beziehen** können. Unten sehen Sie die Beziehung zwischen dem zweiten Artikel und der Kategorie, in der er sich befindet.

Wenn Sie eine SQL-Abfrage schreiben, um Informationen über einen Artikel abzurufen, und Informationen aus einer anderen Tabelle einbeziehen wollen (z.B. den Namen der Kategorie, in der er sich befindet), ist der Artikel das primäre Thema der Abfrage. Daher wird die Tabelle `article` als **linke Tabelle** bezeichnet.

Ziehen Sie zusätzliche Daten über den Artikel aus einer zweiten Tabelle (z.B. der Tabelle `category`), wird diese als **rechte Tabelle** bezeichnet.

Eine `JOIN`-Klausel beschreibt, wie sich die Werte zueinander verhalten.

Artikel								
id	title	summary	content	created	category_id	member_id	image_id	published
1	Systemic Bro...	Brochure...	This bro...	2023-01-01 12:00:00	1	1	1	1
2	Forecast	Handbag...	The handba...	2023-01-01 12:00:00	2	2	2	1
3	Swimming Pool	Architect...	Initiativ...	2023-01-01 12:00:00	3	1	3	1

Kategorie		
id	name	description
1	Print	Inspiring graphic design
2	Digital	Powerful pixels
3	Illustration	Hand-drawn visual storytelling

Bislang haben die Abfragen in diesem Kapitel Daten aus jeweils einer Tabelle in der Datenbank abgerufen. Wenn ein JOIN verwendet wird, um Daten aus mehr als einer Tabelle abzurufen, geben Sie den Namen der Tabelle an, in der sich eine Spalte befindet, sowie den Spaltennamen. Verwenden Sie dazu

- den Namen der Tabelle, in der sich die Daten befinden,
- gefolgt von einem Punkt,
- gefolgt von dem Spaltennamen.

Die nachstehende Abfrage wählt alle Artikeltitel und Zusammenfassungen aus der Tabelle article aus und ruft auch den Namen der Kategorie, zu der jeder Artikel gehört, aus der Tabelle category ab.

1. Dem SELECT-Befehl folgen die Namen der Spalten, aus denen die Daten zurückgegeben werden sollen.

2. Auf den FROM-Befehl folgt der Name der linken Tabelle (das primäre Thema der Abfrage). In diesem Fall ist es die Tabelle article.

3. Nach der JOIN-Klausel folgt der Name der richtigen Tabelle (die Tabelle, die die zusätzlichen Informationen enthält). In diesem Fall ist es die Tabelle category.

Dann teilt die Verknüpfung der Datenbank den Namen einer Spalte sowohl in der linken als auch in der rechten Tabelle mit, deren Werte übereinstimmen sollen.

Verwenden Sie dazu das Schlüsselwort ON, gefolgt von

- der Spalte in der linken Tabelle, die einen Fremdschlüssel enthält,
- dem Symbol für Gleichheit und
- der Spalte in der rechten Tabelle, die einen Primärschlüssel enthält.

```
SELECT article.title, article.summary, category.name
  FROM article
  JOIN category ON article.category_id = category.id;
```

FREMDSCHLÜSSEL

PRIMÄRSCHLÜSSEL

Die nachstehende Ergebnismenge zeigt die ersten drei Datenzeilen, die der Ergebnismenge hinzugefügt werden (die vollständige Ergebnismenge würde alle Artikel enthalten).

Hinweis: Die Spaltennamen in der Ergebnismenge verwenden keine Tabellennamen, da die Daten aus diesen Tabellen entnommen und in einer einzigen Ergebnismenge zusammengefasst wurden.

Ergebnismenge (zeigt die ersten 3 von 24 Zeilen)		
title	summary	name
Milk Beach Website	Website for music series	Digital
Wellness App	App for health facility	Digital
Stargazer Website and App	Website and app for music festival	Digital

Um eine einzelne Zeile oder eine Teilmenge der Zeilen auszuwählen, kann eine Suchbedingung nach dem JOIN hinzugefügt werden. Die nachstehende Abfrage würde zum Beispiel nur die Details von Artikeln zurückgeben, die sich in der Kategorie Print befinden.

Der Suchbedingung können auch Klauseln folgen, die die Ergebnisse ordnen, einschränken und überspringen, wenn sie der Ergebnismenge hinzugefügt werden (wie in früheren Beispielen in diesem Kapitel gezeigt).

```
SELECT article.title, article.summary, category.name
  FROM article
  JOIN category ON article.category_id = category.id
 WHERE category.id = 1;
```

WIE JOINS BEI FEHLENDEM DATEN FUNKTIONIEREN

Wenn die Datenbank versucht, einen JOIN auf einer Datenzeile durchzuführen, aber einige Daten fehlen, können Sie angeben, ob die verfügbaren Daten zur Ergebnismenge hinzugefügt werden sollen oder ob die Zeile übersprungen und nicht zur Ergebnismenge hinzugefügt werden soll.

Stellen Sie sich vor, Sie möchten Daten für jedes Bild abrufen, das für einen Artikel hochgeladen wurde. Sie könnten einen JOIN verwenden, um die Bilddaten abzurufen (analog zum JOIN auf der vorigen Seite, um den Artikeltitel und den Namen der zugehörigen Kategorie abzurufen).

Die Spalte `image_id` der Tabelle `article` ist ein Fremdschlüssel, da ihr Wert der Primärschlüssel der Tabelle `image` ist, die das Bild für den Artikel enthält. Es gibt jedoch einen entscheidenden Unterschied. Während jeder Artikel zu einer Kategorie gehören muss (die Datenbank erzwingt dies durch eine sogenannte Bedingung, die Sie auf S. 431 kennengelernten), muss ein Artikel kein Bild aufweisen.

Wenn für einen Artikel kein Bild hochgeladen wird, enthält die Spalte `image_id` der Tabelle `article` den Wert NULL.

In der nachstehenden Tabelle `article` hat die Spalte `image_id` eines der Artikel den Wert NULL, weil es kein Bild für den Artikel gibt. Dies bedeutet, dass der JOIN keine entsprechenden Bilddaten in der Tabelle `image` finden würde.

Auf der rechten Seite sehen Sie zwei Arten von JOIN, mit denen Sie angeben können, ob die Abfrage den Rest der gefundenen Daten zur Ergebnismenge hinzufügen oder ob sie die gesamte Datenzeile überspringen soll, weil das entsprechende Bild nicht gefunden wurde.

article									
id	title	summary	content	created	category_id	member_id	image_id	published	
4	Walking Birds	Artwork ...	The brie...	2021-02-12	3	3	4	1	
5	Sisters	Editorial...	The arti...	2021-02-27	3	3	NULL	0	
6	Micro-Dunes	Photogra...	This pho...	2021-03-03	1	1	6	1	

image		
id	file	alt
4	birds.jpg	Collage of two birds
6	micro-dunes.jpg	Photograph of tiny sand dunes

INNER JOIN

Ein **innerer Join** fügt der Ergebnismenge Daten hinzu, wenn die Datenbank über *alle* Daten verfügt, um den Join durchzuführen. Um einen inneren Join zu erstellen, verwenden Sie entweder die JOIN- oder die INNER JOIN-Klausel.

```
SELECT article.id, article.title, image.file  
FROM article  
JOIN image ON article.image_id = image.id;
```

Wenn diese Abfrage für die Tabellen auf der linken Seite ausgeführt würde, würde der Artikel mit der id 5 nicht zur Ergebnismenge hinzugefügt, da seine Spalte image_id den Wert NULL hat (die Verknüpfung kann also nicht erstellt werden).

Ergebnismenge (zeigt die ersten 5 von 23 Zeilen)		
id	title	file
1	Systemic Brochure	systemic-brochure.jpg
2	Forecast	forecast.jpg
3	Swimming Pool	swimming-pool.jpg
4	Walking Birds	birds.jpg
6	Micro-Dunes	micro-dunes.jpg

LINKER ÄUSSERER JOIN

Ein linker äußerer Join fügt alle angeforderten Daten aus der linken Tabelle zur Ergebnismenge hinzu. Er verwendet dann NULL für alle Werte, die er nicht aus der rechten Tabelle erhalten kann. Um einen linken äußeren Join zu erstellen, verwenden Sie entweder die LEFT JOIN- oder LEFT OUTER JOIN-Klausel.

Wenn diese Abfrage in den Tabellen auf der linken Seite ausgeführt wird, wird der Titel des Artikels mit der id 5 zur Ergebnismenge hinzugefügt, aber der Wert für die Spalte file erhält den Wert NULL, da keine entsprechenden Daten für das Bild gefunden werden.

```
SELECT article.id, article.title, image.file  
FROM article  
LEFT JOIN image ON article.image_id = image.id;
```

Ergebnismenge (zeigt die ersten 5 von 24 Zeilen)		
id	title	file
1	Systemic Brochure	systemic-brochure.jpg
2	Forecast	forecast.jpg
3	Swimming Pool	swimming pool.jpg
4	Walking Birds	birds.jpg
5	Sisters	NULL

DATEN AUS MEHREREN TABELLEN ABRUFEN

Auf einen SELECT-Befehl können mehrere JOIN-Klauseln folgen, um Daten aus mehr als zwei Tabellen abzurufen.

Um Daten aus mehreren Tabellen abzurufen, können Sie mehr als eine JOIN-Klausel hinzufügen:

- Geben Sie nach der SELECT-Anweisung die Namen aller Spalten an, aus denen Sie Daten abrufen möchten (mit dem Tabellennamen, einem Punkt und dann dem Spaltennamen).
- Verwenden Sie eine JOIN-Klausel, um die Beziehung zwischen den Daten in den einzelnen Tabellen anzugeben.

Die folgende Abfrage ruft Daten über einen Artikel aus drei Tabellen ab: article, category und image.

Die Tabelle `article` ist die linke Tabelle. Sie enthält den Titel und die Zusammenfassung der Artikel.

Die Tabelle `category` enthält den Namen der Kategorie, der ein Artikel jeweils angehört. Da jeder Artikel zu einer Kategorie gehören muss, wird die JOIN-Klausel verwendet.

Die Tabelle `image` enthält den Dateinamen und den Alt-Text des Bilds, das für die einzelnen Artikel verwendet wird. Da nicht jeder Artikel ein Bild haben muss, wird mit einer LEFT JOIN-Klausel sichergestellt, dass alle verfügbaren Daten zur Ergebnismenge hinzugefügt werden.

```
SELECT article.title, article.summary,  
       category.name,  
       image.file, image.alt  
  FROM article  
    JOIN category  ON article.category_id = category.id  
  LEFT JOIN image ON article.image_id      = image.id  
 ORDER BY article.id ASC;
```

Ergebnismenge (zeigt die ersten 3 von 24 Zeilen)

title	summary	name	file	alt
Systemic Brochure	Brochure design for...	Print	systemic-brochure.jpg	Brochure...
Forecast	Handbag illustration...	Illustration	forecast.jpg	Illustrati...
Swimming Pool	Architecture magazine...	Photography	swimming-pool.jpg	Photograph...

MEHRERE JOINS VERWENDEN

SQL

section_c/c11/joins.sql

```
① [SELECT article.id, article.title,
      category.name,
      image.file, image.alt
  ②   FROM article
  ③   JOIN category ON article.category_id = category.id
  ④   LEFT JOIN image ON article.image_id = image.id
  ⑤   WHERE article.category_id = 3
  ⑥     AND article.published = 1
  ⑦   ORDER BY article.id DESC;
```

Ergebnismenge				
id	title	name	file	alt
21	Stargazer	Illustration	stargazer-masc...	Illustrat...
17	Snow Search	Illustration	snow-search.jpg	Illustrat...
10	Polite Society...	Illustration	polite-society...	Mural for...
5	Sisters	Illustration	NULL	NULL
4	Walking Birds	Illustration	birds.jpg	Collage...
2	Forecast	Illustration	forecast.jpg	Illustrat...

Probieren Sie es: Holen Sie sich die gleichen Daten von den Artikeln, die sich in der Kategorie mit der id 2 befinden.

Probieren Sie es: Fügen Sie den Vor- und Nachnamen des Autors aus der member-Tabelle hinzu.

Die Abfrage auf der linken Seite ruft Daten über mehrere Artikel ab, die einer bestimmten Kategorie angehören.

1. Nach der SELECT-Anweisung folgen die Namen der Spalten, die der Ergebnismenge hinzugefügt werden sollen. Die Daten werden aus den Tabellen article, category und image abgerufen.
2. Die FROM-Klausel gibt an, dass die linke Tabelle die article-Tabelle ist.
3. Die erste Verknüpfung gibt an, dass die Daten in der category-Tabelle aus der Zeile stammen sollen, deren Spalte id denselben Wert hat wie die Spalte category_id der article-Tabelle.
4. Die zweite Verknüpfung gibt an, dass die Daten in der Tabelle image aus der Zeile ausgewählt werden sollen, deren Spalte id denselben Wert hat wie die Spalte image_id der Tabelle article. Dies ist ein LEFT JOIN, sodass NULL für fehlende Daten verwendet wird.
5. Die WHERE-Klausel schränkt die Ergebnisse auf Datenzeilen ein, bei denen die Tabelle article in der Spalte category_id den Wert 3 und in der Spalte published den Wert 1 hat.
6. Die ORDER BY-Klausel steuert die Reihenfolge, in der die Ergebnisse zur Ergebnismenge hinzugefügt werden, indem die Artikel-IDs in absteigender Reihenfolge verwendet werden.

ALIASE

Tabellenaliase machen Abfragen, die Joins verwenden, leichter lesbar.
Spaltenaliase geben Spaltennamen in der Ergebnismenge an.

In komplexen SQL-Abfragen, bei denen Joins Daten aus mehreren Tabellen auswählen, können Sie jedem Tabellennamen einen Alias geben.

Ein Tabellenalias ist wie ein Kürzel für einen Tabellennamen und reduziert die Textmenge in der Abfrage.

Ein Tabellenalias wird nach den Befehlen FROM oder JOIN erstellt. Verwenden Sie nach dem Tabellennamen den Befehl AS und geben Sie dann einen Alias für diese Tabelle an. An allen anderen Stellen der Abfrage wird dann der Alias anstelle des vollständigen Tabellennamens verwendet.

```
SELECT t1.column1, t1.column2, t2.column3  
FROM table1 AS t1  
JOIN table2 AS t2 ON t1.column4 = t2.column1;  
                  └      └                  └  
          ALIAS ERSTELLEN  ALIAS FÜR TABELLE 1  ALIAS FÜR TABELLE 2
```

Die Namen der Spalten in der Ergebnismenge werden normalerweise von den Namen der Spalten in den Tabellen übernommen, aus denen die Daten gezogen wurden.

Mit Spaltenaliasen können Sie den Namen einer Spalte in der Ergebnismenge ändern. Sie können auch einer Spalte, die das Ergebnis einer COUNT()-Funktion enthält, einen Namen geben.

Um einen Spaltenalias zu erstellen, verwenden Sie nach der Angabe des Namens einer Spalte, aus der Sie Daten abrufen wollen, den Befehl AS und den Namen, den die Spalte in der Ergebnismenge haben soll.

Oder verwenden Sie nach der Funktion COUNT() den Befehl AS und einen Spaltennamen für die Abfrage in der Ergebnismenge.

DATENBANKSPALTE	ALIAS FÜR ERGEBNISMENGE	COUNT-FUNKTION	ALIAS FÜR ERGEBNISMENGE
SELECT column1 AS newname1 FROM table;		SELECT COUNT(*) AS members FROM members;	

ALIASNAMEN FÜR SPALTENNAMEN VERWENDEN

SQL

section_c/c11/table-alias.sql

```
SELECT a.id, a.title,
       c.name,
       i.file, i.alt
    FROM article      AS a
  JOIN category     AS c  ON a.category_id = c.id
 LEFT JOIN image    AS i  ON a.image_id    = i.id
 WHERE a.category_id = 3
   AND a.published   = 1
 ORDER BY a.id DESC;
```

Diese Abfrage ruft dieselben Daten ab wie das vorherige Beispiel, aber sie gibt Aliase für die Tabellennamen nach den Befehlen FROM und JOIN an:

- a für die Tabelle article
- c für die Tabelle category
- i für die Tabelle image

Die Aliasnamen werden dann anstelle der vollständigen Tabellennamen nach dem SELECT-Befehl und nach den WHERE-, AND- und ORDER BY-Klauseln verwendet.

Probieren Sie es: Ändern Sie die Aliasnamen in:

- art für die article-Tabelle
- cat für die category-Tabelle
- img für die image-Tabelle

Ergebnismenge

id	title	name	file	alt
21	Stargazer	Illustration	stargazer-masc...	Illustrat...
17	Snow Search	Illustration	snow-search.jpg	Illustrat...
10	Polite Society...	Illustration	polite-society...	Mural for...
5	Sisters	Illustration	NULL	NULL
4	Walking Birds	Illustration	birds.jpg	Collage...
2	Forecast	Illustration	forecast.jpg	Illustrat...

SQL

section_c/c11/column-alias.sql

```
SELECT forename AS firstname, surname AS lastname
   FROM member;
```

In diesem Beispiel werden die Werte aus den Spalten forename und surname der Tabelle member ausgewählt und mithilfe von Aliasen mit neuen Spaltennamen in der Ergebnismenge versehen.

Probieren Sie es: Zählen Sie die Anzahl der Artikel in der Tabelle article und verwenden Sie einen Alias, um die Spalte articles aufzurufen.

Ergebnismenge

firstname	lastname
Ivy	Stone
Luke	Wood
Emiko	Ito

SPALTEN KOMBINIEREN & ALTERNATIVEN FÜR NULL

`CONCAT()` fügt Daten aus zwei Spalten in eine Spalte der Ergebnismenge ein. `COALESCE()` gibt einen Wert an, der zu verwenden ist, wenn eine Spalte `NULL` enthält.

Mit der SQL-Funktion CONCAT() werden Werte aus zwei oder mehr Spalten gezogen und die Werte in einer einzigen Spalte in der Ergebnismenge verbunden (oder konkateniert).

Oft wird eine Zeichenkette zwischen den Werten der beiden Spalten eingefügt, um sie zu trennen. Wenn Werte aus zwei Spalten verbunden werden, wird mit einem Alias der Name für diese Spalte in der Ergebnismenge angegeben.

Wie bei jeder Funktion trennt ein Komma die Argumente, die die Funktion CONCAT() zusammenfügt, um den neuen Wert zu erstellen. Im Folgenden werden die Werte aus zwei Spalten zusammengefügt, und die Daten in den beiden Spalten werden durch ein Leerzeichen getrennt.

Wenn der Wert einer Spalte NULL ist, werden die Werte in den anderen Spalten ebenfalls als NULL behandelt.

```
WERT AUS SPALTE 1 TEXT VERBINDELN WERT AUS SPALTE 2  
SELECT CONCAT(column1, ' ', column2) AS newname  
FROM table;  
          |           |           |  
      KOMMA   KOMMA  ALIAS
```

Wenn Sie wissen, dass ein Wert in einer Spalte NULL sein kann, können Sie die SQL-Funktion COALESCE() verwenden, um:

- den Name einer anderen Spalte, deren Wert an ihrer Stelle verwendet werden kann, anzuzeigen (wenn dieser Wert nicht NULL ist, wird er stattdessen verwendet)
 - einen Standardwert anzuzeigen, der zu verwenden ist, wenn alle angegebenen Spalten den Wert NULL haben

Wenn eine Zeile zur Ergebnismenge hinzugefügt wird und der Wert in der ersten Auswahlspalte NULL ist, wird der Wert in der zweiten Auswahlspalte geprüft. Wenn die Werte in allen alternativen Spalten ebenfalls NULL sind, wird der Standardwert verwendet.

Wenn die Funktion COALESCE() verwendet wird, müssen Sie einen Alias für den Spaltennamen in der Ergebnismenge angeben, da die Daten aus mehreren Spalten stammen können.

```
SELECT COALESCE(column1, column2, default) AS newname  
      FROM table;
```

1. DATEN-
AUSWAHL 2. DATEN-
AUSWAHL DEFAULT ALIAS

CONCAT UND COALESCE

SQL

section_c/c11/concat.sql

```
SELECT CONCAT(forename, ' ', surname) AS author  
FROM member;
```

Ergebnismenge

author
Ivy Stone
Luke Wood
Emiko Ito

SQL

section_c/c11/coalesce.sql

```
SELECT COALESCE(picture, forename, 'friend') AS profile  
FROM member;
```

Ergebnismenge

profile
ivy.jpg
Luke
emi.jpg

In diesem Beispiel werden mit der Funktion CONCAT() die Werte aus den Spalten forename und surname in der Spalte author in der Ergebnismenge zusammengeführt.

Zwischen den Werten in den beiden Spalten wurde ein Leerzeichen eingefügt, damit zwischen dem Vor- und dem Nachnamen ein Leerzeichen erscheint.

Probieren Sie es: Fügen Sie die E-Mail-Adresse zu den ausgewählten Werten hinzu und nennen Sie den Alias author_details.

In diesem Beispiel werden mit der SQL-Funktion COALESCE() alternative Werte angegeben, die verwendet werden können, wenn der Wert in der Spalte profile der member-Tabelle NULL ist (weil der Benutzer kein Profilbild hochgeladen hat). Hier sucht die SELECT-Anweisung nach:

- einem Wert in der Spalte picture der member-Tabelle.
- Wenn der Wert NULL ist, wird der Wert in forename verwendet.
- Wenn dieser Wert NULL ist, wird stattdessen der Text friend angezeigt.

Ein Alias gibt den Namen an, den diese Spalte in der Ergebnismenge haben soll. Hier heißt sie profile.

Probieren Sie es: Wenn der Benutzer kein Bild angegeben hat, wird stattdessen der Standardwert placeholder.png verwendet.

ARTIKELABFRAGEN FÜR DAS BEISPIEL-CMS

Das CMS verwendet SQL-Abfragen, um Informationen über einen einzelnen Artikel und die Zusammenfassungen einer Gruppe von Artikeln anzuzeigen. Diese Abfragen vereinen die Techniken, die Sie in diesem Kapitel kennengelernt haben.

1. Die folgende SQL-Abfrage ruft Daten über einen einzelnen Artikel aus allen vier Datenbanktabellen ab.

Auf die SELECT-Anweisung folgen die Namen der Spalten, aus denen die Daten abgerufen werden sollen.

2. Mit Spaltenaliasen werden dem Kategorienamen, dem Dateinamen und dem Alt-Text des Bilds neue Spaltennamen in der Ergebnismenge gegeben.

3. Die Funktion CONCAT() verbindet den Vor- und Nachnamen des Mitglieds, das den Artikel geschrieben hat, und ein Spaltenalias besagt, dass die Ergebnismenge den Namen in der Spalte author speichern soll.

4. Die linke Tabelle ist die article-Tabelle. Drei Joins zeigen die Beziehungen zu den Daten in anderen Tabellen.

Nach den FROM- und JOIN-Befehlen wird jedem Tabellennamen ein Tabellenalias zugeordnet. Jedes Mal, wenn die Abfrage eine Datenspalte angibt, werden diese Aliasnamen anstelle der vollständigen Tabellennamen verwendet.

5. Die WHERE-Klausel gibt die id des abzurufenden Artikels an. Sie wird nur zurückgegeben, wenn die Spalte published den Wert 1 hat.

section_c/c11/article.sql

SQL

```
① SELECT a.title, a.summary, a.content, a.created, a.category_id, a.member_id,
②      c.name      AS category,
③      CONCAT(m.forename, ' ', m.surname) AS author,
④      i.file      AS image_file,
⑤      i.alt       AS image_alt

⑥      [ FROM article      AS a
⑦      JOIN category     AS c  ON a.category_id = c.id
⑧      JOIN member       AS m  ON a.member_id   = m.id
⑨      LEFT JOIN image    AS i  ON a.image_id    = i.id
⑩      WHERE a.id        = 22
⑪      AND a.published   = 1;
```

Ergebnismenge

title	summary	content	created	category_id	member_id	category	author	image_file	image_alt
Polite...	Poster...	These...	2021-0...	1	1	1	Ivy St...	polite-so...	Photogra...

1. Die folgende SQL-Abfrage ruft zusammenfassende Informationen über alle Artikel in einer bestimmten Kategorie ab, wobei Daten aus allen vier Datenbanktabellen verwendet werden.

Nach der SELECT-Anweisung folgen die Namen der Spalten, aus denen die Daten abgerufen werden sollen.

2. Wie im Beispiel auf der linken Seite geben Aliase in der Ergebnismenge den Kategorienamen, der Bilddatei und dem Alt-Text neue Spaltennamen.

Die Funktion CONCAT() wird erneut aufgerufen, um den Vor- und Nachnamen des Autors zu verbinden.

3. Die Joins sind identisch mit dem Beispiel auf der linken Seite.

4. Die WHERE-Klausel gibt die ID der Kategorie an, in der sich die Artikel befinden sollen, und dass der Artikel veröffentlicht sein muss. Die Ergebnisse werden dann nach der Artikel-ID in absteigender Reihenfolge geordnet.

SQL

section_c/c11/article-list.sql

```
① SELECT a.id, a.title, a.summary, a.category_id, a.member_id,
       c.name      AS category,
       CONCAT(m.forename, ' ', m.surname) AS author,
       i.file      AS image_file,
       i.alt       AS image_alt

 ②   FROM article    AS a
 JOIN category   AS c  ON a.category_id = c.id
 JOIN member     AS m  ON a.member_id   = m.id
 LEFT JOIN image  AS i  ON a.image_id   = i.id

 ③ WHERE a.category_id = 1
 ④   AND a.published   = 1
      ORDER BY a.id DESC;
```

Ergebnismenge								
id	title	summary	category_id	member_id	category	author	image_file	image_alt
24	Travel Guide	Book de...	1	1	Print	Ivy Stone	feathervi...	Two page...
22	Polite Society	Poster...	1	1	Print	Ivy Stone	polite-so...	Photogra...
20	Chimney Busin...	Station...	1	2	Print	Luke Wood	chimney-c...	Business...
14	Milk Beach Al...	Packagi...	1	1	Print	Ivy Stone	milk-beac...	Vinyl LP...
12	The Ice Palace	Book co...	1	2	Print	Luke Wood	the-ice-p...	The Ice...
1	Systemic Broc...	Brochure...	1	2	Print	Luke Wood	systemic-...	Brochure...

Probieren Sie es: Wählen Sie dieselben Daten zu jedem Artikel aus, verwenden Sie jedoch eine andere WHERE-Klausel, um die Artikel auszuwählen, die von einem einzelnen Mitglied der Website geschrieben wurden.

Probieren Sie es: Wählen Sie dieselben Daten zu jedem Artikel aus, verwenden Sie jedoch die LIMIT-Klausel, um die sechs zuletzt veröffentlichten Artikel aus der Datenbank (in einer beliebigen Kategorie) abzurufen.

Probieren Sie es: Wählen Sie dieselben Daten zu jedem Artikel aus, verwenden Sie jedoch die LIKE-Klausel, um die Daten zu den Artikeln abzurufen, deren Titel den Begriff design enthält.

DATEN IN DIE DATENBANK EINFÜGEN

Der SQL-Befehl `INSERT INTO` fügt eine neue Datenzeile in eine Tabelle ein. Es können jeweils nur Daten in eine einzelne Tabelle eingefügt werden.

Mit dem Befehl `INSERT INTO` wird die Datenbank angewiesen, Daten in eine einzelne Tabelle einzufügen. Er wird gefolgt von

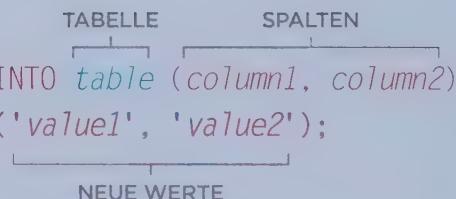
- dem Namen der Tabelle, zu der Sie die Daten hinzufügen möchten, und
- Klammern mit den Namen der Spalten, zu denen Sie die Daten hinzufügen wollen.

Nach dem `VALUES`-Befehl folgen Klammern, die die Werte enthalten, die Sie den Spalten hinzufügen möchten.

Die Werte müssen in der gleichen Reihenfolge erscheinen, in der die Spalten angegeben wurden. Zeichenketten sollten in Anführungszeichen stehen, Zahlen nicht.

WO DIE DATEN EINGEFÜGT WERDEN

HINZUZUFÜGENDE WERTE → `VALUES ('value1', 'value2');`



In der Beispieldatenbank ist die `id`-Spalte jeder Tabelle der Primärschlüssel für diese Tabelle, sodass jede Zeile einen eindeutigen Wert in der `id`-Spalte benötigt.

Um sicherzustellen, dass der Wert für die `id`-Spalte eindeutig ist, wird sie mit der MySQL-Funktion **auto-increment** erstellt. Diese erzeugt eine Zahl für die Spalte und stellt sicher, dass sie eindeutig ist, indem sie jedes Mal um 1 erhöht wird, wenn eine neue Zeile zur Tabelle hinzugefügt wird.

Da dieser Wert von der Datenbank erstellt wird, müssen Sie beim Hinzufügen einer neuen Zeile zu einer Tabelle weder den Spaltennamen `id` noch einen Wert für diese Spalte angeben.

Vier weitere Spalten haben **Standardwerte**, d. h. Sie müssen beim Hinzufügen einer Datenzeile keinen Wert für sie angeben. Der Standardwert für ...

- die Spalte `created` der Tabelle `article` sind das Datum und die Uhrzeit, zu der die Zeile der Datenbank hinzugefügt wurde.
- die Spalte `image_id` der Tabelle `article` ist `NULL`. Wenn kein Bild angegeben wird, enthält diese Spalte `NULL`.
- die Spalte `published` der Tabelle `article` ist `0`. Wenn diese Spalte nicht auf `1` gesetzt ist, ist der Artikel nicht veröffentlicht.
- die `join`-Spalte der Tabelle `member` sind das Datum und die Uhrzeit, zu der die Zeile zur Datenbank hinzugefügt wurde.

SQL

section_c/c11/insert-1.sql

```
INSERT INTO category (name, description, navigation)
VALUES ('News', 'Latest news from Creative Folk', 0);
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	Latest news from Creative Folk	0

Der SQL-Code fügt der Tabelle category die Kategorie News hinzu.

Der SQL-Code muss einen Wert für die Spalten name, description und navigation angeben. Für die Spalte id sollte kein Wert angegeben werden, da die Datenbank diese mithilfe der Auto-Inkrement-Funktion hinzufügt.

Die neue Zeile wird in der Tabelle category auf der linken Seite hervorgehoben.

SQL

section_c/c11/insert-2.sql

```
INSERT INTO image (file, alt)
VALUES ('bicycle.jpg', 'Photo of bicycle'),
       ('ghost.png', 'Illustration of ghost'),
       ('stamp.jpg', 'Polite Society stamp');
```

image		
id	file	alt
22	polite-society-posters.jpg	Photograph of three posters...
23	golden-brown.jpg	Photograph of the interior...
24	featherview.jpg	Two pages from a travel boo...
25	bicycle.jpg	Photo of bicycle
26	ghost.png	Illustration of ghost
27	stamp.jpg	Polite Society stamp

Dieses Beispiel fügt der Tabelle image drei Zeilen hinzu, wobei jede Zeile Details zu einem anderen Bild enthält.

Für jedes Bild muss der SQL-Code den Dateinamen und den Alt-Text angeben. Für die id-Spalte sollte kein Wert angegeben werden, da die Datenbank diese mithilfe der Auto-Inkrement-Funktion hinzufügt.

Die Werte für jede Zeile werden in Klammern gesetzt, genau wie beim Hinzufügen einer Zeile zur Datenbank.

Jeder Satz von Klammern wird durch ein Komma getrennt. Nach der letzten Datenzeile steht ein Semikolon (kein Komma).

Die neuen Zeilen wurden in der Tabelle image hervorgehoben.

Wenn phpMyAdmin nur 25 Datenzeilen anzeigt, gibt es eine Option oberhalb der Ergebnismenge, um alle Daten in der Tabelle anzuzeigen.

Die zusätzlichen Daten, die diese Beispiele der Datenbank hinzufügen, werden in den übrigen Beispielen dieses Kapitels gelöscht.

Sie müssen alle Beispiele in der Reihenfolge ausführen, in der sie im Buch erscheinen. Sonst erhalten Sie Fehlermeldungen.

DATEN IN DER DATENBANK AKTUALISIEREN

Mit dem UPDATE-Befehl von SQL können Sie die Daten in der Datenbank aktualisieren. Der SET-Befehl gibt die zu aktualisierenden Spalten und ihre neuen Werte an. Die WHERE-Klausel steuert, welche Zeile(n) der Tabelle aktualisiert werden sollen.

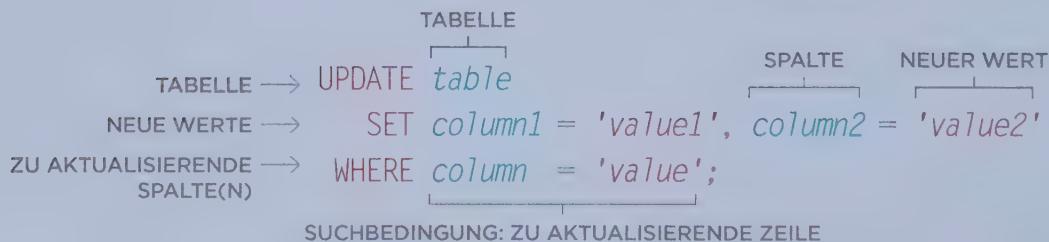
Mit dem Befehl UPDATE teilen Sie der Datenbank mit, dass Sie Daten in der Datenbank aktualisieren wollen. Es folgt der Name der Tabelle(n), die Sie aktualisieren möchten.

Danach werden mit dem Befehl SET die zu aktualisierenden Spalten und die neuen Werte für sie angegeben. Sie müssen nur die Namen und Werte der Spalten angeben, die Sie aktualisieren möchten (die anderen Spalten behalten die bereits vorhandenen Werte).

Mit der WHERE-Klausel wird festgelegt, welche Zeilen aktualisiert werden sollen, genau wie bei der Abfrage bestimmter Datenzeilen aus der Datenbank (wenn sie nicht verwendet wird, wird jede Zeile der Tabelle aktualisiert).

Trifft die Suchbedingung auf mehr als eine Zeile zu, wird jede übereinstimmende Zeile mit denselben Werten aktualisiert.

Um Daten in mehreren Tabellen gleichzeitig zu aktualisieren, würde man in der SQL-Anweisung einen JOIN-Befehl verwenden.



Oft wollen Sie nur eine Zeile auf einmal aktualisieren. In solchen Fällen würde man in der WHERE-Klausel den Primärschlüssel der Tabelle verwenden, um anzugeben, welche Zeile aktualisiert werden soll. In der Beispieldatenbank ist der Primärschlüssel für jede Tabelle der Wert in der Spalte i.d.

Um mehrere Zeilen in einer Tabelle zu aktualisieren, verwenden Sie eine Suchbedingung, die mehr als eine Zeile auswählt. Um beispielsweise alle Artikel eines Autors auszublenden, würde der Wert in der Spalte `published` auf 0 aktualisiert und die Suchbedingung `id` des Autors angeben.

```
UPDATE category
    SET name = 'Blog', navigation = 1
    WHERE id = 5;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	Blog	Latest news from Creative Folk	1

```
UPDATE category
    SET navigation = 0
    WHERE navigation = 1;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	0
2	Digital	Powerful pixels	0
3	Illustration	Hand-drawn visual storytelling	0
4	Photography	Capturing the moment	0
5	Blog	Latest news from Creative Folk	0

EINE DATENBANK SICHERN:

Da eine SQL-Anweisung viele Zeilen in der Datenbank aktualisieren kann, ist es ratsam, eine Sicherungskopie der Datenbank zu erstellen, bevor neue Abfragen ausgeführt werden.

Wenn die SQL-Abfrage versehentlich mehr Daten betrifft als vorgesehen, kann die Sicherung verwendet werden, um die ursprünglichen Daten vor der Ausführung der Abfrage wiederherzustellen.

So erstellen Sie ein Backup Ihrer Datenbank in phpMyAdmin:

1. Wählen Sie die Datenbank aus.
2. Klicken Sie auf die Registerkarte **Export**.
3. Verwenden Sie die gezeigten Optionen und klicken Sie auf **Go**. Es wird SQL generiert, das Sie dann in einer Textdatei speichern sollten. Dies ist dieselbe Datei wie die zum Erstellen der Datenbank verwendete.

Der SQL-Code auf der linken Seite aktualisiert die Zeile, die im letzten Beispiel der Tabelle category hinzugefügt wurde. Sie funktioniert nur mit dieser Zeile, weil die WHERE-Klausel vorschreibt, dass die Kategorie die id 5 haben muss.

Sie ändert den Wert in der Spalte name in Blog und den Wert in der Spalte navigation in 1.

Die aktualisierte Zeile ist in der Tabelle category auf der linken Seite hervorgehoben.

Der SQL-Code auf der linken Seite aktualisiert jede Zeile in der Tabelle category, in der die Spalte navigation den Wert 1 hat (weil die WHERE-Klausel jede Zeile angibt, in der navigation = 1 ist).

Dadurch wird der Wert in der Spalte navigation auf 0 gesetzt, sodass nicht mehr alle Kategorien in der Navigationsleiste angezeigt werden.

Manchmal möchten Sie den Benutzern die Möglichkeit geben, mehrere Zeilen in einer Tabelle zu ändern, aber dieses Beispiel zeigt auch, wie wichtig es ist, dass Ihr SQL nur die gewünschten Zeilen in einer Tabelle aktualisiert.

Probieren Sie es: Verwenden Sie einen SQL-Befehl in phpMyAdmin, um alle Kategorien wieder anzuzeigen.

Hinweis: Sie müssen die Kategorien wieder einschalten, damit sie in den folgenden Kapiteln wieder angezeigt werden.

DATEN AUS DER DATENBANK LÖSCHEN

Mit dem SQL-Befehl `DELETE` werden eine oder mehrere Zeilen aus einer Tabelle entfernt. Der `FROM`-Befehl gibt die Tabelle an, aus der die Daten entfernt werden sollen. Die `WHERE`-Klausel gibt an, welche Zeile(n) der Tabelle gelöscht werden soll(en).

Sie können eine oder mehrere Zeilen aus einer Tabelle gleichzeitig löschen. Verwenden Sie zunächst den Befehl `DELETE FROM`, gefolgt von dem Namen der Tabelle, aus der Sie Daten löschen möchten.

Geben Sie dann mit einer Suchbedingung an, welche Zeilen gelöscht werden sollen (andernfalls werden alle Datenzeilen der Tabelle gelöscht). Um eine Zeile auszuwählen, kann die Bedingung die Spalte mit einem Primärschlüssel angeben.



Wenn die `WHERE`-Klausel mit mehr als einer Datenzeile in der Tabelle übereinstimmt, wird jede der Zeilen aus der Tabelle entfernt.

Mit dem Befehl `DELETE` können Sie keine Werte in einzelnen Spalten der Datenbank löschen. Stattdessen verwenden Sie `UPDATE` und setzen den Wert für die Spalte auf `NULL`.

```
DELETE FROM category
WHERE id = 5;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	Blog	Latest news from Creative Folk	1

Die SQL-Anweisung auf der linken Seite löscht die Zeile aus der Tabelle `category`, in der die Spalte `id` den Wert 5 hat, d. h. die in den vorherigen Beispielen hinzugefügte Kategorie `Blog`.

Die hervorgehobene Zeile wird aus der Tabelle entfernt.

Wenn die WHERE-Klausel mit mehr als einer Datenzeile übereinstimmt, werden alle übereinstimmenden Datenzeilen aus der Tabelle gelöscht.

```
DELETE FROM category
WHERE navigation = 1;
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1

FÜHREN SIE DIESES BEISPIEL NICHT AUS

Achten Sie unbedingt darauf, dass Sie mit einem DELETE-Befehl nicht mehr Daten löschen, als Sie wollen. Die SQL-Anfrage auf der linken Seite löscht zum Beispiel alle Kategorien, die in der Navigation der Website verwendet werden.

Wenn Sie ein Backup der Datenbank erstellen, bevor Sie eine neue SQL-Anweisung ausführen, die Daten aus der Datenbank löscht, stellen Sie sicher, dass Sie über eine Kopie der Daten verfügen, falls die Abfrage nicht die beabsichtigte Aktion ausführt.

EINDEUTIGKEITS-BEDINGUNGEN

In manchen Spalten sollten die Werte eindeutig sein. Zum Beispiel sollten keine zwei Artikel denselben Titel haben, keine zwei Kategorien denselben Namen und keine zwei Mitglieder dieselbe E-Mail-Adresse.

Wenn der Wert in einer Spalte eindeutig sein soll, aber zwei Zeilen denselben Wert in dieser Spalte haben, wird dies als **Dublette** bezeichnet.

Um zu verhindern, dass Zeilen in der Datenbank denselben Wert in einer Spalte haben, kann MySQL angewiesen werden, eine **Eindeutigkeitsbedingung** anzuwenden (diese wird so genannt, weil sie die in dieser Spalte zulässigen Werte einschränkt, damit Ihre Eindeutigkeit sichergestellt ist).

Eindeutigkeitsbedingungen sollten nur im Bedarfsfall verwendet werden, da die Datenbank jedes Mal, wenn Daten hinzugefügt oder bestehende Daten aktualisiert werden, alle anderen Zeilen in dieser Spalte überprüfen muss, um sicherzustellen, dass der Wert nicht bereits existiert. Dies erfordert mehr Rechenleistung und verlangsamt auch die Datenbank.

Im Folgenden sehen Sie, wie Sie in phpMyAdmin eine Eindeutigkeitsbedingung für Kategoriennamen hinzufügen können, um sicherzustellen, dass keine zwei Kategorien denselben Namen haben.

1. Wählen Sie die Datenbank `phpbook - 1` und dann die Tabelle `category` aus dem Menü auf der linken Seite.

2. Wählen Sie die Registerkarte **Structure**.

3. Jede Zeile in der darunter liegenden Tabelle steht für eine Spalte in der Datenbank. Klicken Sie in der Zeile, die für die Namensspalte steht, auf das Drop-down-Menü **More**.

4. Klicken Sie auf den Link **Unique**.

Wenn nun eine SQL-Anweisung versuchen würde, eine Kategorie mit demselben Namen wie eine bereits vorhandene hinzuzufügen oder zu aktualisieren, würde die Datenbank einen Fehler ausgeben. Wie Sie mit dieser Situation umgehen können, erfahren Sie auf S. 491.

The screenshot shows the phpMyAdmin interface for the 'category' table. The left sidebar lists databases and tables. The main area shows the table structure with columns: #, Name, Type, Collation, Attributes, Null, Default, Comments, Extra, and Action. The 'name' column is selected, indicated by a red circle with the number 3. The 'Action' dropdown for this column is open, showing options: Change, Drop, More, Change, Drop, More, Change, Primary, Change, Unique, Primary, Unique, Index, Fulltext, Spatial, and Distinct values. A red circle with the number 4 highlights the 'Unique' option in the dropdown. Other columns shown are 'id' (int(11)), 'description' (varchar(254)), and 'navigation' (tinyint(1)).

In der Beispieldatenbank gibt es drei Tabellen, die jeweils eine Spalte enthalten, für die eine Eindeutigkeitsbedingung erforderlich ist, um sicherzustellen, dass die Werte in dieser Spalte alle unterschiedlich sind.

Dies sind: die Spalte `title` in der Tabelle `article`, die Spalte `name` in der Tabelle `category` und die Spalte `email` in der Tabelle `member`.

FREMDSCHLÜSSEL-BEDINGUNGEN

Wenn es eine Beziehung zwischen zwei Tabellen gibt, prüft eine **Fremdschlüssel-Bedingung**, ob der Wert eines Fremdschlüssels ein gültiger Primärschlüssel in einer anderen Tabelle ist.

Drei Spalten in der Tabelle `article` verwenden Fremdschlüssel:

- `category_id` die ID der Kategorie, in der sich der Artikel befindet
 - `member_id` die Kennung des Mitglieds, das den Artikel verfasst hat
 - `image_id` die ID des Bilds für diesen Artikel
- Die Tabelle `article` verwendet Fremdschlüssel-Bedingungen, um:
- sicherzustellen, dass alle zu diesen Spalten hinzugefügten Werte ein Primärschlüssel in der entsprechenden Tabelle sind (andernfalls erzeugt die Datenbank einen Fehler).
 - zu verhindern, dass eine Kategorie, ein Mitglied oder ein Bild gelöscht wird, wenn sein Primärschlüssel als Fremdschlüssel in der Tabelle `article` verwendet wird.

So fügen Sie einer Tabellenspalte eine Fremdschlüssel-Bedingung hinzu:

1. Wählen Sie die Tabelle mit dem Fremdschlüssel aus.
2. Markieren Sie das Kästchen für die Spalte mit dem Fremdschlüssel.
3. Klicken Sie auf More und auf Index, um einen Index zu der Spalte hinzuzufügen (also eine Kopie der ausgewählten Spalten in der Tabelle, die die Suche in der Tabelle beschleunigt. Dies sollte mit Bedacht eingesetzt werden, da zusätzlicher Speicherplatz beansprucht und die Datenbank verlangsamt werden kann).
4. Wählen Sie die Ansicht Relation.
5. Fügen Sie einen Namen für die Bedingung hinzu.
6. Wählen Sie die Spalte mit dem Fremdschlüssel.
7. Wählen Sie die Tabelle und die Spalte aus, die den Primärschlüssel enthalten. Klicken Sie dann auf Save (nicht abgebildet).

The screenshot shows two instances of the phpMyAdmin interface. The top instance displays the 'article' table structure, where the 'category_id' column is being configured as a foreign key. The bottom instance shows the 'category' table structure, where the 'category_id' column is identified as the primary key for the foreign key constraint. The process involves navigating between tables, selecting columns, and defining constraints through the 'Relation view' and 'Foreign key constraints' sections.

ZUSAMMENFASSUNG

STRUCTURED QUERY LANGUAGE

- SQL wird für die Kommunikation mit Datenbanken verwendet.
- SELECT gibt die Datenspalten an, die aus einer Datenbank abgerufen werden sollen. Die Daten werden dann zu einer Ergebnismenge hinzugefügt.
- CREATE-, UPDATE- und DELETE-Befehle werden zum Erstellen, Aktualisieren oder Löschen von Datenzeilen verwendet.
- FROM gibt an, mit welcher Tabelle gearbeitet werden soll.
- WHERE legt fest, mit welchen Datenzeilen gearbeitet werden soll.
- Joins beschreiben Beziehungen zwischen mehreren Tabellen.
- Ein Primärschlüssel ist eine Spalte mit einem eindeutigen Wert, um jede Zeile zu identifizieren. Der Wert kann mithilfe der Auto-Inkrement-Funktion von MySQL erstellt werden.
- Ein Fremdschlüssel ist eine Spalte, die den Primärschlüssel einer anderen Tabelle speichert und deren Beziehung beschreibt.
- Bedingungen verhindern doppelte Einträge und stellen sicher, dass ein Fremdschlüssel mit einem Primärschlüssel in einer anderen Tabelle übereinstimmt.

12

DATEN
AUS DER
DATENBANK
ABRUFEN

Dieses Kapitel zeigt, wie PHP Daten aus einer Datenbank abrufen und auf einer Seite anzeigen kann. Sie erfahren auch, wie Sie mit einer PHP-Datei mehrere Seiten einer Website anzeigen können.

Im letzten Kapitel wurden SQL-Abfragen (die Datenabfragen in einer Datenbank durchführen) vorgestellt und es wurde gezeigt, wie die Datenbank eine Ergebnismenge mit den angeforderten Daten erstellt. In diesem Kapitel erfahren Sie, wie Sie mit SQL und PHP Daten aus der Datenbank abrufen und in einer Variablen speichern, damit Sie sie in einer Seite verwenden können.

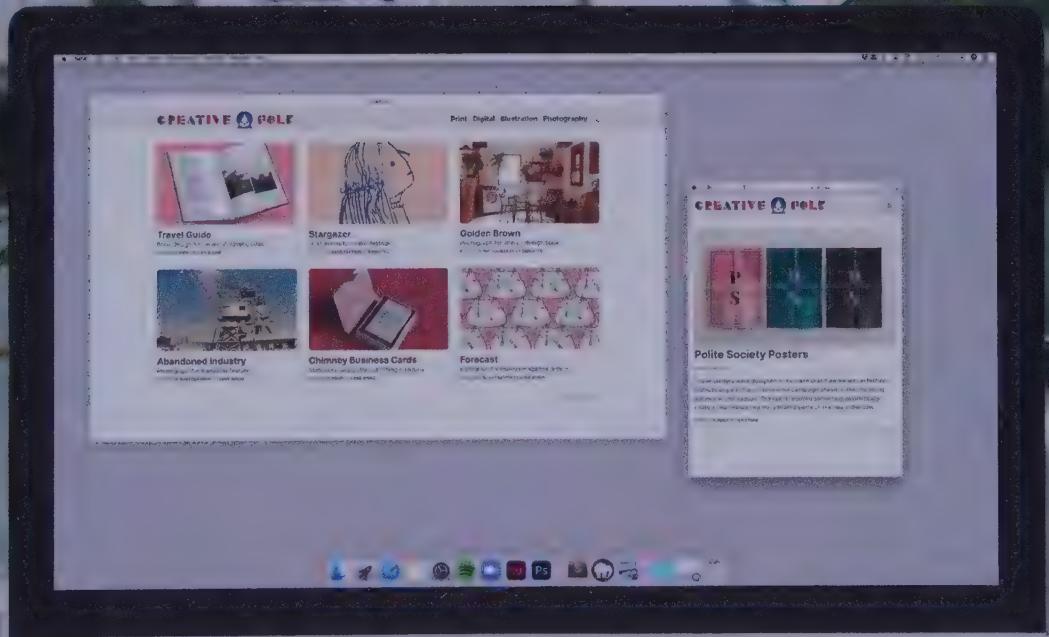
Um dies zu erreichen, verfügt PHP über eine Reihe von eingebauten Klassen, die **PHP Data Objects** (PDO) genannt werden.

- Zunächst wird ein Objekt mit der PDO-Klasse erstellt, um die Verbindung zur Datenbank zu verwalten. Dieses Objekt muss sich mit der Datenbank verbinden, bevor es Daten von ihr abrufen kann, so wie ein FTP-Programm sich mit einem FTP-Server verbindet, bevor Sie Dateien abrufen können, oder ein E-Mail-Programm sich mit einem Mail-Server verbindet, um E-Mails abzurufen.
- Als Nächstes wird mit der Klasse PDOStatement ein Objekt erstellt, das die SQL-Anweisung repräsentiert, die die Datenbank ausführen soll. Sie rufen die Methoden eines PDOStatement-Objekts auf, um die entsprechende SQL-Anweisung auszuführen und die Daten aus der von der Datenbank erzeugten Ergebnismenge abzurufen.

Im größten Teil dieses Kapitels wird jede Datenzeile in der Ergebnismenge durch ein Array dargestellt und in einer Variablen zur Verwendung in der PHP-Seite gespeichert.

Wenn die in der Datenbank gespeicherten Daten von Besuchern zur Verfügung gestellt wurden, müssen sie bereinigt werden, bevor sie auf einer Seite angezeigt werden, um das Risiko eines Cross-Site-Scripting-Angriffs zu vermeiden (siehe S. 244–247).

Am Schluss des Kapitels lernen Sie, wie PDO Daten als Objekte statt als Arrays zur Verfügung stellen kann.



VERBINDUNG ZUR DATENBANK

Ein **Datenquellenname** oder **DSN** ist eine Variable, die die fünf Daten enthält, die PDO benötigt, um die Datenbank zu finden und eine Verbindung zu ihr herzustellen. Mit dem DSN wird nun ein PDO-Objekt mithilfe der eingebauten PDO-Klasse erstellt.

Um einen DSN zu erstellen, werden fünf Daten in Variablen gespeichert. Die Daten in diesen fünf Variablen werden dann zusammengefügt, um einen DSN zu erstellen, der in einer sechsten Variablen gespeichert wird (im Folgenden heißt diese Variable \$dsn).

\$type enthält den Typ der Datenbank. Dies ist erforderlich, da PDO mit vielen Datenbanktypen arbeiten kann. Für MySQL und MariaDB verwenden Sie den Wert mysql.

\$server enthält den Hostnamen des Servers, auf dem die Datenbank gehostet wird. Verwenden Sie für diesen Wert:

- localhost, wenn er sich auf demselben Server wie der Webserver befindet (z.B. wenn Sie MAMP oder XAMPP verwenden).
- die IP-Adresse oder den Domänennamen des Servers, auf dem die Datenbank liegt, wenn es sich nicht um denselben Server handelt.

\$db enthält den Namen der Datenbank, zu der eine Verbindung hergestellt werden soll. Die in diesem Abschnitt verwendete Datenbank ist phpbook-1.

\$type = 'mysql'; // Typ der Datenbanksoftware					
\$server = 'localhost'; // Hostname					
\$db = 'phpbook-1'; // Name der Datenbank					
\$port = '8889'; // Port 3006 für XAMPP verwenden					
\$charset = 'utf8mb4'; // UTF-8-Kodierung, 4 Byte					
\$dsn = "\$type:host=\$server;dbname=\$db;port=\$port;charset=\$charset";	PRÄFIX	HOSTNAME	DATENBANK-NAME	PORTNUMMER	ZEICHENKODIERUNG

\$port enthält die Portnummer für die Datenbank. MAMP verwendet normalerweise Port 8889 und XAMPP Port 3306.

\$charset ist die Zeichenkodierung, in der die Daten an die Datenbank gesendet werden, und die Kodierung, die für die Rücksendung der Daten verwendet werden soll. Diese ist auf utf8mb4 eingestellt.

Mit diesen fünf Werten wird dann der DSN erstellt und in \$dsn gespeichert (die doppelten Anführungszeichen sorgen dafür, dass die Variablennamen durch ihre Werte ersetzt werden, siehe S. 52). Die DSN-Syntax ist sehr präzise; sie darf keine zusätzlichen Leerzeichen oder andere Zeichen enthalten.

- Das Präfix gibt an, um welche Art von Datenbank es sich handelt. Es wird von einem Doppelpunkt gefolgt.
- Dann folgen vier Name/Wert-Paare.
- Jedes Paar wird durch ein Semikolon getrennt.
- Auf den Namen folgt ein =-Symbol, dann der zu verwendende Wert (der jeweils in einer der fünf gerade erstellten Variablen gespeichert wird).

Sobald der DSN in einer Variablen gespeichert wurde, können Sie ein PDO-Objekt erstellen, um die Verbindung zwischen dem Code in der PHP-Datei und der Datenbank zu verwalten.

PDO-Objekte werden mit der in PHP integrierten PDO-Klasse erstellt. Das PDO-Objekt benötigt:

- einen DSN (auf der linken Seite dargestellt), um die zu verbindende Datenbank zu finden.
- den Benutzernamen und das Passwort eines Benutzerkontos, mit dem es sich bei der Datenbank anmelden kann (wie man Benutzerkonten erstellt, haben Sie auf S. 394–395 gesehen).

Im folgenden Code enthält die Variable:

- \$username den Benutzernamen für ein Konto.
- \$password das Passwort für dieses Konto.

Bei der Erstellung eines PDO-Objekts können Sie auch Optionen festlegen, die steuern, wie es mit der Datenbank arbeiten soll. Im Folgenden werden diese Optionen im Array \$options gespeichert.

1. Die Option PDO::ATTR_ERRMODE steuert, wie mit Fehlern, die das PDO-Objekt auslöst, umgegangen wird. Die Einstellung PDO::ERRMODE_EXCEPTION weist PDO an, im Falle eines Fehlers ein Ausnahmeobjekt unter Verwendung der eingebauten PDOException-Klasse zu werfen. Diese Option muss für alle Versionen vor PHP 8 gesetzt werden (sonst würde PDO keine Fehler auslösen), aber in PHP 8 ist dies der Standard-Fehlermodus geworden und kann weggelassen werden.

2. Die Option PDO::ATTR_DEFAULT_FETCH_MODE teilt PDO mit, wie jede Zeile einer Ergebnismenge im PHP-Code verfügbar gemacht werden soll. Die Einstellung PDO::FETCH_ASSOC gibt an, dass jede Zeile der Ergebnismenge als assoziatives Array gespeichert werden soll.

3. Die Einstellung

PDO::ATTR_EMULATE_PREPARES schaltet den sogenannten Emulationsmodus ein/aus. In diesem Buch wird sie auf false gesetzt, um sicherzustellen, dass alle Integer-Datentypen in der Datenbank als int-Datentypen an den PHP-Code zurückgegeben werden. Ist sie auf true gesetzt, wird jeder zurückgegebene Wert als String behandelt.

```
DATENBANK [ $username = 'enter-your-username';
NUTZERAC- COUNT [ $password = 'enter-your-password';
    $options = [
        ①      PDO::ATTR_ERRMODE           => PDO::ERRMODE_EXCEPTION,
        ②      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        ③      PDO::ATTR_EMULATE_PREPARES   => false,
    ];
];
```

Sobald der DSN, die Details des Benutzerkontos und die Optionen in Variablen gespeichert sind, können Sie ein PDO-Objekt mithilfe eines Konstruktors wie jedes andere Objekt erstellen.

```
$pdo = new PDO($dsn, $username, $password, $options);
```

VARIABLE KLASSEN- NAME DSN NUTZERNAME PASSWORT OPTIONEN

Hinweis: Die fünf Werte, die der DSN benötigt, um die Datenbank zu finden und eine Verbindung zu ihr herzustellen, können in den DSN eingefügt werden, anstatt zuerst in Variablen gespeichert zu werden.

Das PDO-Objekt wird in der Variablen \$pdo gespeichert. Da PDO eine integrierte Klasse ist, muss die Klassendefinition nicht in die Seite aufgenommen werden.

Es ist jedoch einfacher, diese Werte zu bearbeiten oder zu ändern, wenn sie in separaten Variablen gespeichert sind, und es ist weniger wahrscheinlich, dass Fehler auftreten, da die Syntax des DSN sehr präzise ist.

DATENBANK-VERBINDUNGEN IN EINEM INCLUDE

In datenbankgestützten Websites stellen die meisten Seiten eine Verbindung zu einer Datenbank her. Der Code zur Erstellung eines PDO-Objekts, das die Verbindung zur Datenbank verwaltet, wird daher häufig in einer Include-Datei gespeichert.

Die rechts abgebildete Include-Datei erstellt ein PDO-Objekt und speichert es in der Variablen \$pdo. Dadurch kann jede Seite, die mit der Datenbank arbeiten muss, diese Datei einbinden und das PDO-Objekt verwenden, das in der Variablen \$pdo gespeichert ist. Die Vorteile der Platzierung dieses Codes in einer Include-Datei sind folgende:

- Sie müssen nicht auf jeder Seite, die sich mit der Datenbank verbindet, denselben Code wiederholen.
- Muss die Datenbankverbindung aktualisiert werden, muss sie nur in der einen Include-Datei aktualisiert werden und nicht in jeder Seite, die sich mit der Datenbank verbindet.
- Haben Sie eine Test- und eine Live-Version einer Website, ändert sich die Datenbankverbindung nur in einer Datei.

Die Datei database-connection.php im CMS-Ordner des Code-Downloads für dieses Kapitel wird sowohl für die Beispiel-Website als auch für die Beispiele in diesem Kapitel verwendet. Sie müssen daher diese Datei bearbeiten, um die Datenbank zu verwenden, bevor Sie die Beispiele ausführen können.

Um zu prüfen, ob Sie eine Verbindung zur Beispieldatenbank herstellen können, aktualisieren Sie die Variablen in Schritt 1 und 2, um die Werte für Ihre Datenbank zu verwenden. Versuchen Sie dann, die Startseite der Beispielseite zu laden. Wenn PDO eine Verbindung zur Datenbank herstellen kann, wird die Seite angezeigt.

Konnte PDO keine Verbindung herstellen, könnte eine Ausnahmebehandlungsfunktion (siehe S. 371) eine Fehlermeldung anzeigen (bei Problemen mit der Verbindung zur Datenbank lesen Sie den Hinweis zur Fehlerbehebung auf der rechten Seite).

1. Speichern Sie die auf der vorherigen Seite gezeigten DSN-Werte.
2. Der Benutzername und das Passwort des Kontos, das für den Zugriff auf die Datenbank eingerichtet wurde, werden in Variablen gespeichert.
3. Dies **müssen** Werte sein, die Sie für Ihre Datenbank erstellt haben.
4. Die Optionen sind so eingestellt, dass alle Fehler, auf die PDO stößt, eine Ausnahme auslösen, PDO angewiesen wird, jede Datenzeile aus einer Ergebnismenge als Array abzurufen, und sichergestellt wird, dass Ganzzahlen als solche und nicht als Strings zurückgegeben werden.
5. Der DSN wird mit den Daten aus Schritt 1 erstellt.
6. Das PDO-Objekt wird in einem Try-Block erstellt, um zu verhindern, dass die Details des Benutzerkontos angezeigt werden (siehe Schritt 8).
7. Wenn das PDO-Objekt erstellt wird, versucht es automatisch, sich mit der Datenbank zu verbinden. Im Erfolgsfall wird das PDO-Objekt in der Variablen \$pdo gespeichert.
8. Wenn PDO keine Verbindung zur Datenbank herstellen kann, wird eine Ausnahme mithilfe der eingebauten PDOException-Klasse geworfen. Der PHP-Interpreter führt dann den Code im catch-Block aus. Wenn der catch-Block ausgeführt wird, wird das Ausnahmeobjekt in der Variablen \$e gespeichert.
9. Die Ausnahme wird im catch-Block erneut ausgelöst. Dies ist wichtig, denn wenn PDO keine Verbindung zur Datenbank herstellen kann und es keinen Exception-Handler für die Site gibt, würde die Fehlermeldung den Benutzernamen und das Passwort der Datenbank anzeigen. Diese Technik verhindert, dass Benutzernamen und Kennwort angezeigt werden.

```

<?php
$�ype      = 'mysql';           // Datenbanktyp
$server    = 'localhost';        // Server
① $db       = 'phpbook-1';       // Name der Datenbank
$port      = '8889';            // Port meist 8889 in MAMP und 3306 in XAMPP
$charset   = 'utf8mb4';          // UTF-8-Encoding mit 4 Bytes pro Zeichen

② [ $username = 'enter-your-username'; // Nutzername hier eingeben
    $password = 'enter-your-password'; // Passwort hier eingeben

    $options = [                  // Optionen für PDO
        PDO::ATTR_ERRMODE          => PDO::ERRMODE_EXCEPTION,
③     PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES => false,
    ];
];
// AB HIER NICHTS LÖSCHEN
④ $dsn = "$type:host=$server;dbname=$db;port=$port;charset=$charset"; // DSN erstellen
⑤ try {
⑥     $pdo = new PDO($dsn, $username, $password, $options); // PDO-Objekt erstellen
⑦ } catch (PDOException $e) { // Wenn Ausnahme geworfen
⑧     throw new PDOException($e->getMessage(), $e->getCode()); // dann erneut werfen
}

```

ERGEBNIS

CREATIVE FOLK

Print · Digital · Illustration · Photography

Quelle: CreativeFolk



Travel Guide

Book design for series of travel guides
POSTED IN PRINT BY IVY STONE



Golden Brown

Photograph for interior design book
POSTED IN PHOTOGRAPHY BY EMIKO ITO



Polite Society Posters

Poster designs for a fashion label
POSTED IN PRINT BY IVY STONE



DIE VERBINDUNG ZUR DATENBANK

Im Code-Download hat jedes verbleibende Kapitel seine eigene Datei für die Verbindung zur Datenbank. Sie müssen diese Datei aktualisieren, damit der Code eine Verbindung zu Ihrer Datenbank herstellen kann, bevor die Beispiele in diesem Kapitel ausgeführt werden können.

PROBLEMLÖSUNG:

Wenn Sie keine Verbindung zur Datenbank herstellen können, ersetzen Sie Schritt 8 durch die Zeile:
include 'database-troubleshooting.php';
Dann nehmen Sie den eigentlichen Code aus Schritt 8.

VERSCHIEDENE DATEN MIT EINER PHP-DATEI ANZEIGEN

Datenbankgesteuerte Websites verwenden SQL-Abfragen, um Daten aus der Datenbank abzurufen und diese zur Erstellung der angezeigten Webseiten zu verwenden.

Im letzten Kapitel haben Sie gesehen, dass SQL-Abfragen Daten über Folgendes abfragen können:

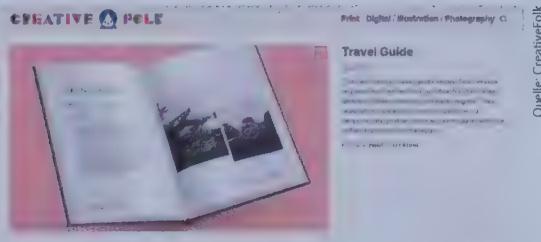
- ein einzelnes Element (einen Artikel oder ein Mitglied)
- eine Gruppe von zusammenhängenden Artikeln (Zusammenfassungen der letzten Artikel oder alle Artikel eines Mitglieds)

In der Beispieleite wird mit der Datei `article.php` jede Artikelseite mit den in der Datenbank gespeicherten Daten angezeigt. Die URL für die Seite gibt in einem Abfragestring an, welchen Artikel die SQL-Abfrage aus der Datenbank ziehen soll. Die Seite `article.php` verwendet PDO, um die Abfrage auszuführen, und die Datenbank erstellt eine Ergebnismenge mit einer Datenzeile, die den Artikel repräsentiert. Diese Daten werden in einem Array gespeichert und dann auf der Seite angezeigt.

Die unten abgebildete Startseite (`index.php`) zeigt mithilfe einer SQL-Abfrage Details zu den letzten sechs Artikeln an, die der Website hinzugefügt wurden.

Wenn ein neuer Artikel zur Website hinzugefügt wird, ruft die Startseite die Details dieses neuen Artikels aus der Datenbank ab und zeigt ihn als neuesten Artikel auf der Startseite an.

Wenn eine SQL-Abfrage eine Reihe von zusammenhängenden Elementen abfragt, z. B. die letzten sechs Artikel, die der Site hinzugefügt wurden (oder alle Artikel, die von einem Mitglied der Site geschrieben wurden), kann die von der Datenbank erstellte Ergebnismenge mehrere Datenzeilen enthalten. Jede Datenzeile in der Ergebnismenge stellt einen Artikel dar und kann in einem Array dargestellt werden. Diese Daten können dann auf der Seite angezeigt werden.



Quelle: CreativeFolk



Quelle: CreativeFolk

Eine einzelne PHP-Datei kann mehrere SQL-Abfragen verwenden, um die für die Erstellung der Seite erforderlichen Informationen aus der Datenbank abzurufen.

Die Kategorieseite (category.php) kann Details zu jeder einzelnen Kategorie der Website anzeigen. Sie zeigt zunächst den Namen und die Beschreibung der Kategorie an, gefolgt von Zusammenfassungen aller Artikel, die zu dieser Kategorie gehören. Dazu werden zwei SQL-Abfragen verwendet:

- Die erste SQL-Abfrage ermittelt den Namen und die Beschreibung der Kategorie (diese Abfrage erzeugt eine Ergebnismenge mit einer Datenzeile).
- Die zweite Abfrage ruft alle Artikel in dieser Kategorie ab (diese Abfrage erzeugt eine Ergebnismenge, die mehrere Datenzeilen enthält – jede Zeile steht für einen neuen Artikel). Wenn der Kategorie ein neuer Artikel hinzugefügt wird, wird dieser automatisch auf der Seite angezeigt.

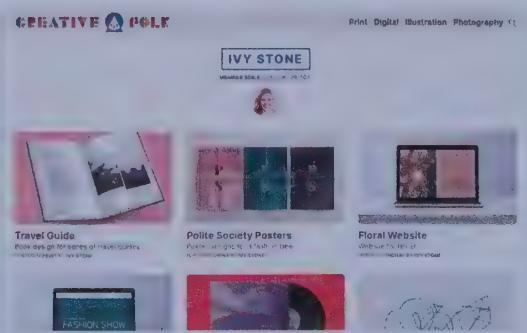
Die Mitgliederseite (member.php) dient dazu, das Profil eines jeden Mitglieds anzuzeigen. Sie zeigt zunächst das Profil des Mitglieds (Name, Bild und Beitrittsdatum), gefolgt von Zusammenfassungen aller Artikel, die das Mitglied veröffentlicht hat. Auch für diese Seite sind zwei Abfragen erforderlich:

Mit der ersten werden der Name, das Profilbild und das Beitrittsdatum des Mitglieds abgefragt (dadurch wird eine Ergebnismenge mit einer Datenzeile erstellt).

Die zweite Abfrage ermittelt alle Artikel, die das Mitglied geschrieben hat (diese Abfrage erzeugt eine Ergebnismenge, die mehrere Datenzeilen enthalten kann – jede Zeile steht für einen neuen Artikel). Wenn ein Mitglied einen neuen Artikel zur Website hinzufügt, wird dieser automatisch auf der Seite angezeigt.



Quelle: CreativeFolk



Quelle: CreativeFolk

DATEN MIT EINER SQL-ABFRAGE ABRUFEN

Wenn eine SQL-Anweisung ausgeführt wird, erstellt die Datenbank eine Ergebnismenge. Jede Datenzeile in der Ergebnismenge kann als assoziatives Array dargestellt werden.

Die folgende SQL-Abfrage ruft den Vor- und Nachnamen eines Mitglieds der Website ab. Sie verwendet eine WHERE-Klausel, um Daten über das Mitglied abzufragen, das die id 1 hat:

```
SELECT forename, surname  
      FROM member  
     WHERE id = 1;
```

Da diese Abfrage nur Daten über ein Mitglied der Website abfragt, enthält die Ergebnismenge nie mehr als eine Datenzeile.

Ergebnisse	
forename	surname
Ivy	Stone

Unten sehen Sie, wie die Datenzeile als assoziatives Array dargestellt werden kann. Die Spaltennamen in der Ergebnismenge werden als Schlüssel im Array verwendet; ihre Werte sind die Werte der jeweiligen Spalte.

```
$member = [  
    'forename' => 'Ivy',  
    'surname'  => 'Stone',  
];
```

Die folgende SQL-Abfrage ermittelt den Vor- und Nachnamen jedes Mitglieds der Website.

```
SELECT forename, surname  
      FROM member;
```

Diese Abfrage erstellt eine Ergebnismenge, die mehrere Datenzeilen enthält.

Ergebnisse	
forename	surname
Ivy	Stone
Luke	Wood
Emiko	Ito

Wenn eine SQL-Abfrage eine Ergebnismenge mit mehreren Datenzeilen erzeugt, wird ein indiziertes Array erstellt. Der Wert jedes Elements in diesem indizierten Array ist ein assoziatives Array, und jedes dieser assoziativen Arrays stellt eine Datenzeile aus der Ergebnismenge dar.

```
$members = [  
    0 => ['forename' => 'Ivy',  
          'surname'  => 'Stone',],  
    1 => ['forename' => 'Luke',  
          'surname'  => 'Wood',],  
    2 => ['forename' => 'Emiko',  
          'surname'  => 'Ito',],  
];
```

Die Methode `query()` des PDO-Objekts führt eine SQL-Abfrage aus und erstellt ein PDOStatement-Objekt, das die von der Datenbank erstellte Ergebnismenge repräsentiert. Die Methoden des PDOStatement-Objekts rufen Daten aus der Ergebnismenge ab.

Die Methode `query()` des PDO-Objekts hat einen Parameter: eine SQL-Abfrage, die die Datenbank ausführen soll.

Wenn die Methode `query()` aufgerufen wird, wird die SQL-Abfrage ausgeführt und ein PDOStatement-Objekt zurückgegeben.

Das PDOStatement-Objekt stellt die Ergebnismenge dar, die die Datenbank für die Abfrage erstellt hat.

Die folgende SQL-Abfrage ermittelt den Vor- und Nachnamen jedes Mitglieds der Website.

Wenn die unten stehende PHP-Anweisung ausgeführt wird, gibt das PDO-Objekt ein PDOStatement-Objekt zurück, das die Vor- und Nachnamen aller Mitglieder der Website enthält. Dieses Objekt wird in der Variablen `$statement` gespeichert.

```
$statement = $pdo->query("SELECT forename, surname FROM member");
```

The diagram shows the code `$statement = $pdo->query("SELECT forename, surname FROM member");`. Brackets below the code group `$pdo` and `query("...")` under the label "OBJEKT". Above the code, a bracket groups "SELECT forename, surname FROM member" under the label "SQL-ABFRAGE". Another bracket groups `query()` under the label "query()-METHODE".

Mit der `fetch()`-Methode des PDOStatement-Objekts wird eine einzelne Datenzeile aus der Ergebnismenge abgerufen.

Diese Datenzeile wird durch ein assoziatives Array dargestellt und in einer Variablen gespeichert, damit der restliche Code der PHP-Seite sie verwenden kann.

```
$member = $statement->fetch();
```

The diagram shows the code `$member = $statement->fetch();`. Brackets below the code group `$statement` and `fetch()` under the label "OBJEKT". Another bracket groups `fetch()` under the label "METHODE".

Wenn die Abfrage eine leere Datensatzmenge erzeugt, gibt die Methode `fetch()` den Wert `false` zurück.

Wenn die Abfrage mehrere Datenzeilen erzeugt hat, sammelt die `fetchAll()`-Methode des PDOStatement-Objekts alle Daten aus der Ergebnismenge in einem indizierten Array. Jedes Element dieses Arrays speichert ein assoziatives Array, das eine Zeile der Daten in der Ergebnismenge repräsentiert.

```
$members = $statement->fetchAll();
```

The diagram shows the code `$members = $statement->fetchAll();`. Brackets below the code group `$statement` and `fetchAll()` under the label "OBJEKT". Another bracket groups `fetchAll()` under the label "METHODE".

Wenn die Abfrage eine leere Datensatzmenge erzeugt, gibt die Methode `fetchAll()` ein leeres Array zurück.

EINE DATENZEILE AUS EINER DATENBANK ABRUFEN

Dieses Beispiel zeigt, wie eine einzige PHP-Datei Daten über jedes einzelne Mitglied der Website anzeigen kann.

1. Die Datei database-connection.php wird in die Seite eingebunden. Sie erstellt das PDO-Objekt, das die Verbindung zur Datenbank verwaltet, und speichert es in der Variablen \$pdo.

2. functions.php wird eingefügt. Sie enthält die Definition für die Funktion html_escape() (S. 247), die mit htmlspecialchars() die reservierten HTML-Zeichen durch Entities ersetzt und so XSS-Angriffe verhindert.

3. Eine SQL-Anweisung wird in der Variablen \$sql gespeichert. Sie ruft den Vor- und Nachnamen des Mitglieds ab, dessen ID 1 ist.

4. Die Methode query() des PDO-Objekts wird aufgerufen. Ihr einziges Argument ist die Variable mit der SQL-Anweisung, die ausgeführt werden soll.

Die Methode query() führt die Abfrage der SQL-Anweisung aus und gibt ein PDOStatement-Objekt mit der Ergebnismenge zurück. Das zurückgegebene PDOStatement-Objekt wird in der Variablen \$statement gespeichert.

5. Die fetch() -Methode des PDOStatement-Objekts ruft die Daten über dieses Mitglied ab und speichert sie als assoziatives Array in der Variablen \$member.

section_c/c12/examples/query-one-row.php

PHP

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ [ $sql      = "SELECT forename, surname
                  FROM member
                  WHERE id = 1;";
④ $statement = $pdo->query($sql);
⑤ $member   = $statement->fetch();
?>
<!DOCTYPE html>
<html> ...
<body>
<p>
⑥    <?= html_escape($member['forename']) ?>
    <?= html_escape($member['surname']) ?>
</p>
</body>
</html>
```

ERGEBNIS

Ivy Stone

6. Der Vorname des Mitglieds wird mit html_escape() ausgeschrieben, um alle in HTML reservierten Zeichen im Vornamen des Mitglieds durch die entsprechenden Entitäten zu ersetzen.

7. Der Nachname des Mitglieds wird mit html_escape() ausgeschrieben.

Probieren Sie es: Ändern Sie in Schritt 3 die SQL WHERE-Klausel, um das Mitglied mit der ID 2 abzufragen. Dadurch wird ein anderes Mitglied der Website angezeigt.

Ändern Sie nun die Klausel, um ein Mitglied mit der ID 4 abzufragen. Es wird eine Fehlermeldung angezeigt, da die Datenbank nur 3 Mitglieder hat.

PRÜFEN, OB DIE ABFRAGE DATEN ZURÜCKGEGEBEN HAT

PHP

section_c/c12/examples/checking-for-data.php

```
<?php
require '../cms/includes/database-connection.php';
require '../cms/includes/functions.php';
$sql      = "SELECT forename, surname
              FROM member
             WHERE id = 4;";
①   $statement = $pdo->query($sql);
②   $member    = $statement->fetch();
③   if (!$member) {
④       include 'page-not-found.php';
}
?>
<!DOCTYPE html>
<html> ...
<body>
<p>
    <?= html_escape($member['forename']) ?>
    <?= html_escape($member['surname']) ?>
</p>
</body>
</html>
```

ERGEBNIS

Sorry! We cannot find that page.

Try the home page or email us hello@eg.link

Hinweis:

page-not-found.php entspricht der Seite auf S. 378–379, führt aber zwei zusätzliche Aufgaben aus: Zunächst setzt sie den HTTP-Antwortcode auf 404 (siehe S. 242).

Nach einer Meldung, die besagt, dass die Seite nicht gefunden wurde, wird mit dem Befehl exit kein weiterer Code mehr ausgeführt (weder in dieser Datei noch in der Datei, die sie enthält).

Wenn eine SQL-Abfrage keine übereinstimmenden Daten findet, wird beim Aufruf der fetch()-Methode des PDOStatement-Objekts der Wert false zurückgegeben.

Würde die Datei dann versuchen, die Daten des Mitglieds anzuzeigen, würde der PHP-Interpreter einen Undefined index-Fehler auslösen, da die Variable \$member den Wert false und kein Array enthält.

Um diese Fehler zu vermeiden, kann die Datei prüfen, ob Daten gefunden wurden. Wenn nicht, kann sie dem Benutzer mitteilen, dass die Seite nicht gefunden werden konnte.

1. Die SQL-Abfrage fragt nach dem Mitglied mit der id 4. (In der Datenbank gibt es kein Mitglied mit der id 4.)
2. Die Methode fetch() wird aufgerufen. Sie gibt den Wert false zurück, der in der Variablen \$member gespeichert wird.
3. Bevor versucht wird, die Daten anzuzeigen, prüft eine if-Anweisung, ob der Wert in \$member falsch ist (mit dem not-Operator siehe S. 54).

Ist dies der Fall, wurde das Mitglied nicht gefunden.

4. Die Datei page-not-found.php wird in die Seite eingefügt, um den Benutzern mitzuteilen, dass die Seite nicht gefunden wurde.

Probieren Sie es: Ändern Sie in Schritt 1 die ID in 2.

MEHRERE ZEILEN AUS EINER DATENBANK ABRUFEN

Dieses Beispiel zeigt, wie eine PHP-Datei Daten über jedes Mitglied der Website abrufen und anzeigen kann. Wenn ein neues Mitglied in die Datenbank aufgenommen wird, zeigt die Seite automatisch dessen Details an.

1. database-connection.php wird eingebunden, um ein PDO-Objekt zu erstellen, das in der Variablen \$pdo gespeichert wird. functions.php wird eingebunden, weil darin die Funktion html_escape() eingeschlossen ist.

2. Die auszuführende SQL-Anweisung wird in der Variablen \$sql gespeichert.

Sie ruft den Vor- und Nachnamen jedes Mitglieds ab.

3. Die Methode query() des PDO-Objekts wird aufgerufen. Ihr Parameter ist die auszuführende SQL-Anweisung.

Die Methode query() führt die SQL-Abfrage aus und gibt ein PDOStatement-Objekt zurück, das die Ergebnismenge enthält, die in der Variablen \$statement gespeichert ist.

4. Die Methode fetchAll() des PDOStatement-Objekts ruft jede Datenzeile aus der Ergebnismenge ab. Sie gibt die Daten als ein indiziertes Array zurück, das in \$members gespeichert wird. Jedes Element dieses Arrays steht für eine Zeile aus der Ergebnismenge. Der Wert eines jeden Elements ist ein assoziatives Array, das ein Mitglied repräsentiert.

section_c/c12/examples/query-multiple-rows.php

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ $sql      = "SELECT forename, surname
   FROM member;";
④ $statement = $pdo->query($sql);
⑤ $members   = $statement->fetchAll();
?>
<!DOCTYPE html>
<html> ...
<body>
⑤ <?php foreach ($members as $member) { ?>
    <p>
        <?= html_escape($member['forename']) ?>
        <?= html_escape($member['surname']) ?>
    </p>
<?php } ?>
</body>
</html>
```

PHP

ERGEBNIS

Ivy Stone

Luke Wood

Emiko Ito

5. Eine foreach-Schleife arbeitet sich durch die Elemente in dem indizierten Array, das in \$members gespeichert ist. Jedes Mal, wenn die Schleife läuft, wird das assoziative Array, das ein Mitglied der Website repräsentiert, in der Variablen \$member gespeichert.

6. Der Vor- und Nachname des Mitglieds wird angezeigt.

HINWEIS: Wenn eine Abfrage keine Daten liefert, gibt fetchAll() ein leeres Array zurück und die Anweisungen in der Schleife werden nicht ausgeführt (sonst würde ein Undefined index-Fehler auftreten).

DATENZEILEN MIT SCHLEIFEN NACHEINANDER ABRUFEN

PHP section_c/c12/examples/query-multiple-rows-while-loop.php

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ $sql      = "SELECT forename, surname
                FROM member;";
④ $statement = $pdo->query($sql);
?>
<!DOCTYPE html>
<html> ...
<body>
④    <?php while ($row = $statement->fetch()) { ?>
        <p>
            <?= html_escape($row['forename']) ?>
            <?= html_escape($row['surname']) ?>
        </p>
    <?php } ?>
</body>
</html>
```

ERGEBNIS

Ivy Stone

Luke Wood

Emiko Ito

Mit einer `while`-Schleife können Sie das `PDOStatement`-Objekt anweisen, jeweils eine Datenzeile aus der Datenbank abzurufen, wie in diesem Beispiel gezeigt.

1. `database-connection.php` und `functions.php` werden eingefügt.

2. Die auszuführende SQL-Anweisung wird in der Variablen `$sql` gespeichert.

3. Die Methode `query()` des `PDO`-Objekts wird aufgerufen, um den SQL-Code auszuführen und das `PDOStatement`-Objekt für die Ergebnismenge zu erzeugen.

4. Die Bedingung der `while`-Schleife ruft die Methode `fetch()` auf. Dadurch wird jeweils eine Datenzeile aus der Ergebnismenge zurückgegeben.

Das Array, das diese Datenzeile repräsentiert, wird in der Variablen `$row` gespeichert, damit es von den Anweisungen innerhalb der Schleife verwendet werden kann.

Sobald die Schleife einmal gelau-
fen ist, ruft sie erneut die Methode `fetch()` auf, ruft automatisch die nächste Datenzeile aus der Ergebnismenge ab und speichert sie in `$row`. Wenn keine weiteren Zeilen in der Ergebnismenge vorhanden sind, gibt die Methode `fetch()` `false` zurück, und die Schleife hört auf zu laufen.

5. Innerhalb der Schleife wird der Inhalt des Arrays mit dem Namen des Mitglieds ausgeschrieben.

HINWEIS: Normalerweise sollte eine Website nicht zu viele Informationen auf einer einzigen Seite anzeigen (sondern die Ergebnisse auf mehrere Seiten aufteilen). Wenn eine Seite jedoch mit großen Datenmengen arbeiten muss (mehr als auf einer typischen Webseite angezeigt werden), sollte dieser Ansatz verwendet werden, um nicht zu viel Speicher zu verbrauchen. Das liegt daran, dass `fetchAll()` alle Daten sammelt und sie in einem Array im Speicher des PHP-Interpreters ablegt, während die `while`-Schleife jeweils nur eine Datenzeile aus der Datenbank abruft.

VERÄNDERLICHE DATEN VERWENDEN

Sobald eine Seite angefordert wird, können unterschiedliche Werte in der SQL-Abfrage verwendet werden, um unterschiedliche Daten aus der Datenbank abzurufen. In solchen Fällen muss die SQL-Abfrage erst **vorbereitet** und dann **ausgeführt** werden.

SCHRITT 1: VORBEREITUNG

SQL-Anweisungen verwenden **Platzhalter**, um Werte darzustellen, die sich bei jeder Ausführung der SQL-Anweisung ändern können.

Ein SQL-Platzhalter verhält sich wie eine Variable, aber sein Name beginnt mit einem Doppelpunkt und nicht mit einem \$-Symbol.

Wenn eine SQL-Abfrage einen Platzhalter enthält, wird nicht die Methode `query()` des PDO-Objekts aufgerufen, um die Abfrage auszuführen, sondern die Methode `prepare()` des PDO-Objekts.

Die `prepare()`-Methode gibt ebenfalls ein `PDOStatement`-Objekt zurück, aber an diesem Punkt steht das `PDOStatement`-Objekt nur für die SQL-Anweisung (nicht die Ergebnismenge).

SCHRITT 2: AUSFÜHRUNG

Anschließend wird die SQL-Anweisung ausgeführt, indem die Methode `execute()` des PDOStatement-Objekts aufgerufen wird.

Sie benötigt ein Argument: ein Array mit den Werten, die als Ersatz für die Platzhalter verwendet werden sollen.

Der Name des Platzhalters und der zu verwendende Wert können als assoziatives Array übergeben werden, wobei:

- key der Name des Platzhalters in der SQL-Abfrage ist (ihm kann ein Doppelpunkt vorangestellt werden, aber dies ist nicht erforderlich)
 - value der Wert ist, mit dem der Platzhalter ersetzt wird (der Wert ist häufig bereits in einer Variablen gespeichert)

```
$sql      = "SELECT forename, surname FROM member WHERE id = :id;";  
$statement = $pdo->prepare($sql);  
$statement->execute(['id' => $id]);
```


PLATZHALTERNAME ZU VERWENDENDER WERT

Die obige SQL-Abfrage enthält den Platzhalter :id.
Die Methode execute() ersetzt :id durch den in
der Variablen \$id gespeicherten Wert. Programmierer
nennen dies eine **vorbereitete oder parametrisierte**
Anweisung.

Fügen Sie **NIEMALS** Werte aus einem Abfragestring oder einem Formular in eine Zeichenfolge ein, um eine SQL-Anweisung zu erstellen (wie unten gezeigt).

Dadurch wird Ihre Website für einen Hack verwundbar, der als SQL-Injection-Angriff bekannt ist. Vorbereitete Anweisungen verhindern dies.

```
(X) $sql = 'SELECT * FROM member WHERE id=' . $id;  
(X) $sql = 'SELECT * FROM member WHERE id=' . $_GET['id'];
```

VERSCHIEDENE DATEN AUF DERSELBEN SEITE ANZEIGEN

PHP

section_c/c12/examples/prepared-statement.php

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ $id      = 1;
④ $sql     = "SELECT forename, surname
              FROM member
              WHERE id = :id;";
⑤ $statement = $pdo->prepare($sql);
⑥ $statement->execute(['id' => $id]);
⑦ $member   = $statement->fetch();
⑧ if (!$member) {
    include 'page-not-found.php';
}
?>
<!DOCTYPE html>
<html> ...
<body>
<p>
    <?= html_escape($member['forename']) ?>
    <?= html_escape($member['surname']) ?>
</p>
</body>
</html>
```

ERGEBNIS

Ivy Stone

Probieren Sie es: In Schritt 2 ändern Sie den in \$id gespeicherten Wert in die Zahl 2.

Speichern und aktualisieren Sie die Seite dann.

Die Seite zeigt die Daten eines anderen Mitglieds an. Wenn \$id auf 4 gesetzt ist, wird page-not-found.php angezeigt.

1. datenbank-connection.php und functions.php werden eingefügt.

2. Die Variable \$id enthält die Ganzzahl 1, die der ID des Mitglieds entspricht, das die Abfrage aus der Datenbank abrufen soll.

3. Die auszuführende SQL-Anweisung wird in der Variablen \$sql gespeichert.

Der Teil der Daten, der sich ändern kann (die ID des abzufragenden Mitglieds), verwendet den Platzhalter :id.

4. Die Methode prepare() des PDO-Objekts wird aufgerufen. Sie gibt ein PDOStatement-Objekt zurück, das die Abfrage repräsentiert. Dieses wird in der Variablen \$statement gespeichert.

5. Die execute()-Methode des PDOStatement-Objekts wird aufgerufen, um die Abfrage auszuführen und die Ergebnismenge zu erzeugen. Ihr Argument ist ein Array, das den Platzhalternamen und den Wert, durch den er ersetzt werden soll, enthält.

6. Mit der fetch()-Methode des PDOStatement-Objekts wird die Datenzeile aus der Ergebnismenge abgerufen und als assoziatives Array in der Variablen \$member gespeichert.

7. Wenn keine Daten zurückgegeben werden, wird dem Benutzer mitgeteilt, dass die Seite nicht gefunden wurde.

WERTE AN EINE SQL-ABFRAGE BINDEN

Die Methoden `bindValue()` und `bindParam()` des `PDOStatement`-Objekts bieten eine weitere Möglichkeit, eine vorbereitete Anweisung zu erstellen und einen Platzhalter in einer SQL-Abfrage zu ersetzen.

Wenn die Methoden `bindValue()` und `bindParam()` verwendet werden, um einen Platzhalter in einer SQL-Abfrage zu ersetzen, sollten sie nach der Methode `prepare()` des `PDOStatement`-Objekts und vor der Methode `execute()` aufgerufen werden. Sie haben drei Parameter:

- den Namen des Platzhalters
- eine Variable, deren Wert den Platzhalter ersetzt
- einen Wert, der den Datentyp des Werts angibt (dies ist nicht erforderlich, wenn der Datentyp ein String ist)

Die folgende Tabelle zeigt die Werte, die im dritten Parameter der Methode `bindValue()` verwendet werden, um den Datentyp des Werts anzugeben, der den Platzhalter in der SQL-Abfrage ersetzt.

DATA TYPE	VALUE
String	<code>PDO::PARAM_STR</code>
Integer	<code>PDO::PARAM_INT</code>
Boolean	<code>PDO::PARAM_BOOL</code>

Der Unterschied zwischen `bindValue()` und `bindParam()` ist der Punkt, an dem der PHP-Interpreter den Wert aus der Variablen erhält und mit ihm den Platzhalter in der SQL-Abfrage ersetzt.

- Mit `bindValue()` ruft der Interpreter den Wert aus der Variablen ab, wenn `bindValue()` aufgerufen wird.
- Mit `bindParam()` ruft der Interpreter den Wert aus der Variablen ab, wenn `execute()` aufgerufen wird.

Wenn sich also der in der Variablen gespeicherte Wert zwischen dem Aufruf von `bindParam()` und `execute()` ändert, wird der aktualisierte Wert verwendet.

Im Folgenden würde der Platzhalter `:id` in der SQL-Abfrage durch den Wert in der Variablen `$id` ersetzt.

Im weiteren Verlauf des Buchs wird die auf der vorigen Seite gezeigte Technik zum Binden von Daten verwendet, da der Datentyp nicht für jeden der Werte angegeben werden muss und weniger Code benötigt wird.

```
PLATZHALTER  
$sql      = "SELECT * FROM member WHERE id = :id;";  
$statement = connection->prepare($sql);  
$statement->bindValue('id', $id, PDO::PARAM_INT);  
              +-----+  
              | PLATZHALTER | VARIABLE | DATENTYP |
```

EINE GANZZAHL AN EINE SQL-ABFRAGE BINDEN

PHP

section_c/c12/examples/bind-value.php

```
<?php
require '../cms/includes/database-connection.php';
require '../cms/includes/functions.php';
$id      = 1;
$sql     = "SELECT forename, surname
            FROM member
           WHERE id = :id;";
$statement = $pdo->prepare($sql);
① $statement->bindValue('id', $id, PDO::PARAM_INT);
② $statement->execute();
$member   = $statement->fetch();
if (!$member) {
    include 'page-not-found.php';
}
?>
<!DOCTYPE html>
<html> ...
<body>
<p>
    <?= html_escape($member['forename']) ?>
    <?= html_escape($member['surname']) ?>
</p>
</body>
</html>
```

ERGEBNIS

Ivy Stone

Dieses Beispiel sieht genauso aus wie das vorherige, aber es verwendet die `bindValue()`-Methode des `PDOStatement`-Objekts, um den Platzhalter in der SQL-Abfrage zu ersetzen.

1. Nachdem die Methode `prepare()` des `PDO`-Objekts aufgerufen wurde, um ein `PDOStatement`-Objekt für die SQL-Abfrage zu erstellen, wird die Methode `bindValue()` des `PDOStatement`-Objekts aufgerufen, um den Platzhalter in der SQL-Abfrage zu ersetzen. Sie verwendet drei Argumente:

- den Namen des Platzhalters in der SQL-Abfrage
- den Namen der Variablen, die den Wert enthält, der für den Platzhalter verwendet werden soll
- `PDO::PARAM_INT`, um anzugeben, dass der Wert eine ganze Zahl ist

2. Die Methode `execute()` des `PDOStatement`-Objekts wird aufgerufen, um die Abfrage auszuführen. Sie benötigt keine Argumente, da der Platzhalter in der SQL-Abfrage bereits durch einen Wert ersetzt wurde.

Probieren Sie es: In Schritt 2 ändern Sie den in `$id` gespeicherten Wert in die Zahl 2. Speichern Sie dann und aktualisieren Sie die Seite. Die Seite zeigt die Daten eines anderen Mitglieds an. Wenn `$id` auf 4 gesetzt ist, wird `page-not-found.php` eingefügt.

EINE EINZELNE DATEI ZUR ANZEIGE VIELER SEITEN VERWENDEN

Wenn eine einzige PHP-Datei verwendet wird, um viele Seiten einer Website anzuzeigen, kann ein Abfragestring an das Ende der URL angehängt werden. Dieser teilt der PHP-Datei mit, welche Daten sie aus der Datenbank abrufen muss.

Eine PHP-Seite kann einen Wert aus einem Abfragestring erfassen und diesen Wert dann in einer SQL-Abfrage verwenden, um anzugeben, welche Daten aus der Datenbank gezogen werden sollen.

`Ivy Stone`

Die Seite member.php verwendet die PHP-Funktion filter_input(), um den Wert aus dem Abfragestring abzurufen. Da die id-Spalten in der Datenbank allesamt Ganzzahlen sind, verwendet die Funktion den Integer-Filter. Der Wert, den sie zurückgibt, wird in der Variablen \$id gespeichert:

- Wenn es sich um eine ganze Zahl handelt, enthält \$id diese ganze Zahl.
- Handelt es sich nicht um eine ganze Zahl, enthält \$id den Wert false.
- Ist id nicht im Abfragestring enthalten, enthält \$id den Wert null.

```
$id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
if (!$id) {
    include 'page-not-found.php';
}
```

Enthält der Abfragestring eine gültige ganze Zahl, kann die Seite weiter versuchen, Daten aus der Datenbank zu ziehen und sie in der Variablen \$member zu speichern. Dann kann eine zweite if-Anweisung prüfen, ob die Mitgliedsdaten gefunden wurden. Wenn nicht, wird die Datei page-not-found.php eingefügt (wodurch die Ausführung der Seite angehalten wird).

Der Link zu member.php enthält einen Abfragestring mit dem Namen id und dem Wert 1. Dieser Wert entspricht der Spalte id in der Tabelle member.

Als Nächstes prüft eine if-Anweisung, ob \$id entweder false oder null ist (weil im Abfragestring keine gültige ganze Zahl verwendet wurde). Dann kann die Seite die Mitgliederdaten nicht aus der Datenbank abrufen, sodass die Datei page-not-found.php eingefügt wird. Diese Datei

- sendet einen HTTP-Antwortcode mit dem Wert 404.
- teilt dem Besucher mit, dass die Seite nicht gefunden werden konnte.
- verwendet den Befehl exit, um die Ausführung des Codes anzuhalten.

Nur wenn die Mitgliederdaten erfolgreich aus der Datenbank abgerufen wurden, wird der Rest der Seite angezeigt.

Um die Daten eines anderen Mitglieds der Website anzuzeigen, würde der Abfragestring die ID der entsprechenden Zeile in der Mitgliedertabelle der Datenbank liefern.

ABFRAGEFOLGEN VERWENDEN, UM DIE RICHTIGE SEITE ANZUZEIGEN

PHP section_c/c12/examples/query-strings.php?id=1

```
<?php
require '../cms/includes/database-connection.php';
require '../cms/includes/functions.php';

① $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
② if (!$id) {                                // Wenn keine ID
③     include 'page-not-found.php';          // Page not found
}

④ [ $sql      = "SELECT forename, surname
                  FROM member
                  WHERE id = :id;";           // SQL-Abfrage
    $statement = $pdo->prepare($sql);        // Vorbereiten
    $statement->execute([':id' => $id]);    // Ausführen
    $member   = $statement->fetch();          // Daten holen

⑤ if (!$member) {                            // Wenn keine Daten
⑥     include 'page-not-found.php';          // Page not found
}
?>
<!DOCTYPE html>
<html> ...
    <body>
        <p>
            <?= html_escape($member['forename']) ?>
            <?= html_escape($member['surname']) ?>
        </p>
    </body>
</html>
```

ERGEBNIS

Ivy Stone

Dieses Beispiel baut auf den vorherigen Beispielen auf und verwendet einen Abfragestring, um der Seite die `id` des anzuseigenden Mitglieds mitzuteilen.

1. Die PHP-Funktion `filter_input()` zieht die `ID` des Mitglieds aus dem Abfragestring. Wenn es eine ganze Zahl ist, wird `$id` auf diese Zahl gesetzt. Wenn nicht, wird sie auf `false` gesetzt. Fehlt sie, wird `$id` auf `null` gesetzt.
2. Eine `if`-Anweisung prüft, ob `$id` den Wert `false` oder `null` hat.
3. Wenn ja, wird `page-not-found.php` eingefügt, da der Abfragestring keine Mitgliedsnummer enthält, die angibt, welches Mitglied angezeigt werden soll.
4. Wenn die Seite noch läuft, wird im Abfragestring eine ganze Zahl für das Mitglied angegeben, so dass die Seite versucht, die Daten dieses Mitglieds aus der Datenbank zu ziehen.
5. Eine weitere `if`-Anweisung prüft, ob `$member` den Wert `false` hat.
6. Wenn ja, wurde kein Mitglied gefunden, und `page_not_found.php` wird eingefügt.
7. Andernfalls wird die HTML-Seite mit den Daten des Mitglieds erstellt.
Probieren Sie es: Ändern Sie die Zahl im Abfragestring auf 4, und es wird `page not found` angezeigt.

DATENBANKDATEN IN HTML-SEITEN ANZEIGEN

Ziehen Sie zunächst die Daten aus der Datenbank und speichern Sie sie in Variablen. Dann verwenden Sie die Daten in diesen Variablen, um die HTML-Seite zu erstellen.

Um Ihren Code leichter lesbar zu machen, sollten Sie in Ihren PHP-Dateien klar trennen zwischen:

- Code, der Daten aus der Datenbank abruft
- Code, der die HTML-Seite erzeugt

FUNKTIONEN

Mit Funktionen wird häufig sichergestellt, dass die Daten korrekt formatiert werden. Mit der Funktion `html_escape()` wurde bereits auf vielen Seiten ein XSS-Angriff verhindert, indem eines der reservierten HTML-Zeichen durch Entities ersetzt wird.

Rechts sehen Sie eine weitere Funktion, die sich in der Datei `functions.php` zu diesem Kapitel befindet. Mit ihr wird sichergestellt, dass die von der Datenbank generierten Daten in einer konsistenten, für den Menschen lesbaren Form formatiert werden.

BEDINGTE ANWEISUNGEN

Eine bedingte Anweisung kann anhand der von der Datenbank zurückgegebenen Daten entscheiden, was der HTML-Code anzeigen soll. Wenn der Benutzer beispielsweise kein Profilbild hochgeladen hat, gibt die Datenbank `NULL` als Dateiname für das Profilbild zurück.

SCHLEIFEN

Wie Sie bereits in mehreren Beispielen gesehen haben, wird mit einer Schleife üblicherweise jede Zeile der Ergebnismenge, die eine Datenbank zurückgegeben hat, durchlaufen.

Auf der rechten Seite ist dies durch eine gepunktete Linie dargestellt.

Der Teil der Datei, der die HTML-Seite erstellt, sollte so wenig PHP-Code wie möglich enthalten; unten sehen Sie die drei am häufigsten benötigten Code-Arten.

Zunächst werden das Datum und die Uhrzeit, die in der Datenbank gespeichert sind, mit der PHP-Funktion `strtotime()` in einen Unix-Zeitstempel umgewandelt. Anschließend erfolgt mit der PHP-Funktion `date()` eine Umwandlung in ein für Menschen lesbares Format.

```
function format_date(string $string):  
string  
{  
    $date = strtotime($string);  
    return date('F d, Y', $date);  
}
```

Das Profilbild kann mit dem Null-Koaleszenz-Operator angezeigt werden; wenn ein Bild angegeben wurde, wird es angezeigt, wenn nicht, wird stattdessen der Platzhalter angezeigt.

```
html_escape($member['picture'] ??  
'blank.png');
```

Auf der rechten Seite sehen Sie, dass mit einer `foreach`-Schleife die gleichen Anweisungen für jedes Mitglied der Website wiederholt werden, für das die Datenbank Details liefert.

IN HTML-SEITEN VERWENDETE DATEN FORMATIEREN

PHP

section_c/c12/examples/formatting-data-in-html.php

```
<?php
    require '../cms/includes/database-connection.php';           // PDO-Obj erstellen
    require '../cms/includes/functions.php';                     // Funktionen
    $sql      = "SELECT id, forename, surname, joined, picture FROM member"; // SQL
    $statement = $pdo->query($sql);
    $members   = $statement->fetchAll();                         // Abfrage ausführen
    $members   = $statement->fetch();
    // Daten holen
?>
<!DOCTYPE html> ...
<body> ...
② <?php foreach ($members as $member) { ?>
    <div class="member-summary">
        " class="profile">
        <h2><?= html_escape($member['forename']) . ' ' . $member['surname'] ?></h2>
        <p>Member since:<br><?= format_date($member['joined']) ?></p>
    </div>
<?php } ?> ...
</body>
```

ERGEBNIS



Ivy Stone

Member since:
January 26, 2021



Luke Wood

Member since:
January 26, 2021



Emiko Ito

Member since:
February 12, 2021

1. Die Seite sammelt Daten über jedes Mitglied der Website.
2. Eine `foreach`-Schleife wiederholt für jedes Mitglied der Website dieselben Anweisungen.

3. Ein Null-Koaleszenz-Operator prüft, ob ein Profilbild angegeben wurde. Wenn ja, wird der Dateiname in einem ``-Tag verwendet. Ist dies nicht der Fall, wird stattdessen die Platzhalterbilddatei `blank.png` angezeigt.

4. Das Datum des Beitritts eines Mitglieds wird mit `format_date()` formatiert. Die Schleife wiederholt dieselben Anweisungen für jedes Mitglied der Website.

EINE FUNKTION ZUM AUSFÜHREN VON SQL-ANWEISUNGEN

Um PDO dazu zu bringen, eine SQL-Abfrage auszuführen und die Ergebnismenge zurückzugeben, sind zwei oder drei Anweisungen erforderlich; wenn Sie eine benutzerdefinierte Funktion schreiben, benötigen Sie nur eine einzige.

Bei einer SQL-Abfrage ohne Parameter führt die Methode `query()` des PDO-Objekts eine SQL-Abfrage aus und gibt ein PDOStatement-Objekt zurück, das die Ergebnismenge repräsentiert:

```
$statement = $pdo->query($sql);
```

Bei einer SQL-Abfrage mit Parametern muss die `prepare()`-Methode des PDO-Objekts aufgerufen werden, um ein PDOStatement-Objekt zu erstellen, das die Abfrage repräsentiert.

Dann muss die `execute()`-Methode des PDOStatement-Objekts aufgerufen werden, um die Abfrage auszuführen:

```
$statement = $pdo->prepare($sql);
$statement->execute($sql);
```

Nach diesen Schritten muss eine der Methoden des PDOStatement-Objekts die Daten aus der Ergebnismenge abrufen.

Die folgende Funktion hat drei Parameter:

- `$pdo` steht für das PDO-Objekt, das zur Verwaltung der Verbindung zur Datenbank verwendet wird.
- `$sql` steht für die SQL-Abfrage, die ausgeführt werden soll.
- `$arguments` ist ein Array von SQL-Parameternamen und ihren Ersetzungswerten; beachten Sie, dass der Standardwert `null` ist, wenn dieses Argument nicht angegeben wird.
- Die Funktion prüft, ob keine Argumente angegeben wurden.
- Wenn nicht, wird die Methode `query()` ausgeführt und das erzeugte PDOStatement-Objekt zurückgegeben.
- Falls angegeben, ruft sie die `prepare()`-Methode auf, um ein PDOStatement-Objekt zu erstellen, ruft die `execute()`-Methode dieses Objekts auf, um die Abfrage auszuführen, und gibt dann das PDOStatement-Objekt zurück.

```
function pdo(PDO $pdo, string $sql, array $arguments = null)
{
    if (!$arguments) {
        return $pdo->query($sql);
    }
    $statement = $pdo->prepare($sql);
    $statement->execute($arguments);
    return $statement;
}
```

Beim Aufruf der benutzerdefinierten pdo()-Funktion wird mit der **Methodenverkettung** die Abfrage ausgeführt, und die Daten werden in einer einzigen Anweisung zurückgegeben.

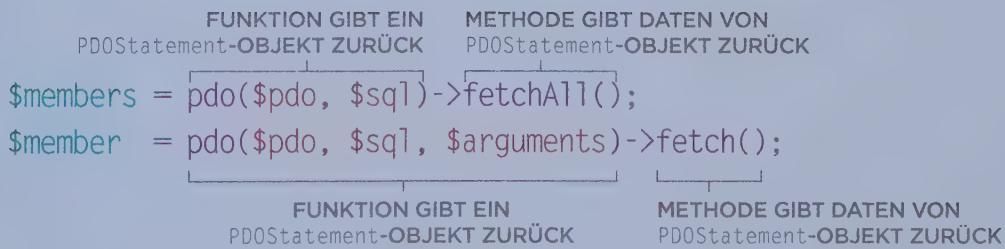
Wenn ein PDOStatement-Objekt erstellt und die SQL-Anweisung ausgeführt wurde, wird eine der folgenden drei Methoden aufgerufen, um die Daten aus der Ergebnismenge abzurufen und in einer Variablen zu speichern:

- fetch() ruft eine einzelne Datenzeile ab.
- fetchAll() ruft mehrere Datenzeilen ab.

fetchColumn() ruft einen Wert aus einer einzelnen Spalte ab.

Wenn eine Funktion oder Methode ein Objekt zurückgibt, können Sie mit der **Methodenverkettung** eine Methode des Objekts aufrufen, das in derselben Anweisung zurückgegeben wird.

Wenn im Folgenden die Funktion pdo() aufgerufen wird, gibt sie ein PDOStatement-Objekt zurück. Nach dem Funktionsaufruf folgen ein Objektorientator und ein Aufruf einer Methode des PDOStatement-Objekts, das zurückgegeben wurde.



Die Definition der Funktion pdo() wurde in die Include-Datei functions.php für dieses Kapitel eingefügt. Sie wird für den Rest dieses und des nächsten Kapitels verwendet.

In Kapitel 14 lernen Sie eine weitere Technik kennen, bei der benutzerdefinierte Klassen verwendet werden, um Objekte zu erstellen, die die in der Datenbank gespeicherten Daten abrufen und ändern.

Wenn Sie Argumente für die execute()-Methode des PDOStatement-Objekts bereitstellen, können Sie ein assoziatives Array nutzen, dessen Schlüssel mit den Namen der Parameter in der SQL-Anweisung übereinstimmen (wie bereits gezeigt), oder Sie können ein indiziertes Array bereitstellen, bei dem die Werte in der gleichen Reihenfolge stehen wie die Platzhalter in der SQL-Anweisung (siehe S. 459).

BENUTZERDEFINIERT PDO-FUNKTION OHNE PARAMETER

Dieses Beispiel zeigt, wie die auf der vorherigen Seite definierte Funktion pdo() Daten aus der Datenbank abruft, wenn eine SQL-Abfrage keine Parameter hat.

1. Die Datei database-connection.php wird eingefügt. Sie erstellt das PDO-Objekt und speichert es in der Variablen \$pdo.
2. functions.php wird eingefügt. Sie enthält die Funktion pdo() (wie auf der vorherigen Seite gezeigt).

3. Die SQL-Abfrage zum Abrufen der Vor- und Nachnamen der einzelnen Mitglieder wird in der Variablen \$sql gespeichert.

4. Die Funktion pdo() wird mit zwei Argumenten aufgerufen:

- dem PDO-Objekt, das in Schritt 1 erstellt wurde
- der SQL-Abfrage, die in Schritt 3 in der Variablen \$sql gespeichert wurde

Dies gibt ein PDOStatement-Objekt zurück, dessen fetchAll()-Methode in derselben Anweisung aufgerufen wird, um alle Daten aus der Ergebnismenge abzurufen. Diese Daten werden als Array in der Variablen \$members gespeichert.

5. Eine foreach-Schleife zeigt die im Array \$members gespeicherten Daten an.

section_c/c12/examples/pdo-function-no-parameters.php **PHP**

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ $sql = "SELECT forename, surname
          FROM member;";
④ $members = pdo($pdo, $sql)->fetchAll();
?>
<!DOCTYPE html>
<html> ...
<body>
<?php foreach ($members as $member) { ?>
    <p>
        <?= html_escape($member['forename']) ?>
        <?= html_escape($member['surname']) ?>
    </p>
<?php } ?>
</body>
</html>
```

ERGEBNIS

Ivy Stone

Luke Wood

Emiko Ito

Probieren Sie es: In Schritt 3 aktualisieren Sie die SQL-Anweisung, um neben dem Namen auch die E-Mail-Adressen der Mitglieder abzurufen. In Schritt 5 zeigen Sie die E-Mail-Adresse nach dem Namen an.

BENUTZERDEFINIERT PDO-FUNKTION MIT PARAMETERN

PHP section_c/c12/examples/pdo-function-with-parameters.php?id=1

```
<?php
require '../cms/includes/database-connection.php';
require '../cms/includes/functions.php';
① $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
if (!$id) {
    include 'page-not-found.php';
}

② [
    $sql = "SELECT forename, surname
           FROM member
          WHERE id = :id;";
③ $member = pdo($pdo, $sql, ['id' => $id])->fetch();

if (!$member) {
    include 'page-not-found.php';
}
?> ...
④ [
    <?= html_escape($member['forename']) ?>
    <?= html_escape($member['surname']) ?>
</p>
```

ERGEBNIS

Ivy Stone

Probieren Sie es: In Schritt 3 ersetzen Sie das assoziative Array, das als dritter Parameter verwendet wird, durch die Variable \$id in eckigen Klammern:

```
$member = pdo($pdo, $sql, [$id]);
```

Dadurch wird der dritte Parameter zu einem indizierten Array (und nicht zu einem assoziativen Array). Wenn diese Technik verwendet wird, müssen die Werte im Array in der gleichen Reihenfolge stehen wie die Platzhalter in der SQL-Anweisung.

Dieses Beispiel zeigt, wie die auf Seite 456 definierte Funktion pdo() Daten aus der Datenbank abruft, wenn eine SQL-Abfrage Parameter verwendet.

1. Der Abfragestring enthält die ID des Mitglieds, das angezeigt werden soll.
2. Die SQL-Abfrage ermittelt den Vor- und Nachnamen eines Mitglieds, dessen ID durch den Platzhalter :id dargestellt wird.
3. Die Funktion pdo() wird mit drei Argumenten aufgerufen:
 - dem PDO-Objekt, das in database-connection.php erstellt wurde
 - die in Schritt 2 in der Variablen \$sql gespeicherte SQL-Abfrage
 - einem assoziativen Array mit dem Namen eines SQL-Platzhalters und dem Wert, den er verwenden soll

HINWEIS: Das Array wird im Argument erstellt und nicht erst in einer Variablen gespeichert:

```
['id' => $id]
```

Dadurch wird ein PDOStatement-Objekt zurückgegeben. Mit seiner fetch()-Methode werden in derselben Anweisung die einzelnen Datenzeilen abgerufen, die die SQL-Abfrage erzeugt. Diese Daten werden in \$member gespeichert.

4. Die in \$member gespeicherten Daten werden auf der Seite ausgegeben.

EIN PAAR PHP-DATEIEN STEUERN EINE GANZE WEBSITE

Die nächsten zwölf Seiten zeigen, dass vier PHP-Dateien genügen, um mehr als fünfzig Seiten der Beispiel-Website anzuzeigen.

Jede dieser vier PHP-Dateien ist in zwei Teile aufgeteilt:

- Zunächst zieht ein Code Daten aus der Datenbank und speichert sie in Variablen.
- Dann werden mit den Daten in diesen Variablen die HTML-Seiten erstellt, die an den Besucher zurückgesendet werden.

Sie können sich diese Dateien wie eine Vorlage vorstellen.

Jedes Mal, wenn eine der Dateien angefordert wird, kann sie verschiedene Daten aus der Datenbank abrufen, in den entsprechenden Teil der HTML-Seite einfügen und dann den HTML-Code an den Browser des Besuchers senden.

index.php

Die Startseite zeigt Zusammenfassungen der letzten sechs Artikel, die auf der Website veröffentlicht wurden.

Wenn ein neuer Artikel in der Datenbank gespeichert wurde, wird diese Seite automatisch aktualisiert, um die Details des neuen Artikels anzuzeigen (und der älteste der sechs Artikel wird nicht mehr angezeigt).

Die Struktur der HTML-Seite bleibt immer gleich, aber der Inhalt, den sie aus der Datenbank zieht, kann sich ändern.

Jede Seite verwendet vier Include-Dateien:

- `database-connection.php` erstellt ein PDO-Objekt, das die Verbindung zur Datenbank verwaltet.
- `functions.php` enthält die Funktionen, die sowohl Daten aus der Datenbank abrufen als auch die Daten formatieren.
- `header.php` enthält die Kopfzeile für jede Seite und erstellt die Navigation.
- `footer.php` enthält die Fußzeile für jede Seite.



Quelle: Creativefolk

DIGITAL

POWERBALL POSTS



Floral Website
Website for florist
POSTED IN DIGITAL BY IVY STONE



Polite Society Website
Website for fashion label
POSTED IN DIGITAL BY IVY STONE



Chimney Press Website
Website for book publisher
POSTED IN DIGITAL BY LUKE WOOD



Responsive Mobile App



18/10/2020 App



18/10/2020 App

Quelle: CreativeFolk

category.php

Diese Datei kann den Titel und die Beschreibung einer beliebigen Kategorie anzeigen, gefolgt von Zusammenfassungen der einzelnen Artikel, die in dieser Kategorie veröffentlicht wurden. Die Struktur der HTML-Seite bleibt gleich, aber die angezeigten Daten können sich ändern.

Ein Abfragestring nach der URL enthält die ID der Kategorie, die aus der Datenbank abgerufen und auf der Seite angezeigt werden soll.

Zum Beispiel:

category.php?id=1

IVY STONE

MEMBER SINCE: JANUARY 20, 2021



Travel Guide
Book design for series of travel guides
POSTED IN PRINT BY IVY STONE



Polite Society Posters
Poster designs for a fashion label
POSTED IN PRINT BY IVY STONE



Floral Website
Website for florist
POSTED IN DIGITAL BY IVY STONE

Quelle: CreativeFolk

member.php

Die Mitgliederdatei kann das Profil jedes Mitglieds der Website anzeigen. Es folgen Zusammenfassungen der Artikel, die sie geschrieben haben.

Ein Abfragestring nach der URL enthält die ID des Mitglieds, die aus der Datenbank abgerufen und auf der Seite angezeigt werden soll.

Zum Beispiel:

mitglied.php?id=1

**Travel Guide**

April 23, 2021

This post selling travel guide series. Featherweight, required a refreshed look and feel for their latest series of books covering the Asian region. They were after a clean and concise solution - a versatile design that could accommodate both the coffee table and the backpack.

POSTED IN PRINT BY IVY STONE

Quelle: CreativeFolk

article.php

Die Artikeldatei kann jeden beliebigen Artikel anzeigen. Das Bild, die Kategorie, der Titel, das Datum, der Inhalt und der Autor sind für jeden Artikel an der gleichen Stelle, aber die angezeigten Daten können sich ändern.

Ein Abfragestring nach der URL enthält die ID des Artikels, die aus der Datenbank abgerufen und auf der Seite angezeigt werden soll.

Zum Beispiel:

article.php?id=1

KOPF- UND FUSSZEILEN

Die Kopfzeile ist für alle Seiten der Website gleich. Anstatt diesen Code in jeder Datei zu wiederholen, wird er daher in der Datei header.php eingefügt.

Bevor diese Datei in eine Seite eingebunden wird, müssen vier Daten (siehe unten) in Variablen gespeichert werden, weshalb es wichtig ist, sich diese Datei zuerst anzusehen.

Die ersten beiden Informationen werden in den HTML-Elementen <title> und <meta>-Beschreibung verwendet.

Die zweiten beiden Variablen werden zur Erstellung der Navigationsleiste verwendet. Die erste enthält eine Reihe von Kategorien, die angezeigt werden sollen; mit der zweiten wird die Kategorie hervorgehoben, wenn der Besucher in dieser Kategorie sucht.

VARIABLE	WERT
\$title	Ein Wert, der im <title>-HTML-Element der Seite angezeigt wird.
\$description	Ein Wert, der im <meta>-Description-Tag der Seite angezeigt wird.
\$navigation	Ein Array mit den Namen und IDs der Kategorien, die in der Navigationsleiste erscheinen.
\$section	Handelt es sich bei der Seite um eine Kategorienseite, steht hier die Kennung der Kategorie, die gerade angezeigt wird. Handelt es sich bei der Seite um eine Artikelseite, wird hier die ID der Kategorie angegeben, in der sich der Artikel befindet. Mit diesen Werten kann die aktive Kategorie in der Navigationsleiste hervorgehoben werden. Für alle anderen Seiten enthält diese Variable eine leere Zeichenkette.

1. Der Inhalt der Variablen \$title wird innerhalb des Elements <title> angezeigt.

2. Der Inhalt der Variablen \$description wird innerhalb des <meta> description-Tags angezeigt.

3. Eine foreach-Schleife arbeitet sich durch jede Kategorie im Array \$navigation.

Jedes Mal, wenn die Schleife läuft, werden die ID und der Name einer Kategorie als assoziatives Array in der Variablen \$link gespeichert.

4. Ein Link zu category.php wird erstellt. Der Abfragestring enthält die ID der Kategorie, um der Datei category.php mitzuteilen, welche Kategorie angezeigt werden soll.

5. Ein ternärer Operator bestimmt, ob diese Kategorie hervorgehoben werden soll.

Die Bedingung prüft, ob der Wert in \$section mit der ID der aktuellen Kategorie in der Schleife übereinstimmt.

Wenn die Werte übereinstimmen, werden class="on" und aria-current="page" zum Link hinzugefügt, um anzudeuten, dass dies die aktuelle Kategorie ist.

6. Der Name der Kategorie wird im Link ausgeschrieben.

7. Nach den Links zu den Kategorienseiten findet sich ein Link zur Suchseite.

8. footer.php enthält nur eine PHP-Anweisung zur Anzeige des aktuellen Jahrs nach einem Copyright-Hinweis.

9. site.js enthält JavaScript für die responsive Navigation der Seite.

①
②

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title><?= html_escape($title) ?></title>
    <meta name="description" content="<?= html_escape($description) ?>">
    <link rel="stylesheet" type="text/css" href="css/styles.css">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link rel="stylesheet"
      href="https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap">
    <link rel="shortcut icon" type="image/png" href="img/favicon.ico">
  </head>
  <body>
    <header>
      <div class="container">
        <a class="skip-link" href="#content">Skip to content</a>
        <div class="logo">
          <a href="index.php"></a>
        </div>
        <nav role="navigation">
          <button id="toggle-navigation" aria-expanded="false">
            <span class="icon-menu"></span><span class="hidden">Menu</span>
          </button>
          <ul id="menu">
            <?php foreach ($navigation as $link) { ?>
            <li><a href="category.php?id=<?= $link['id'] ?>">
              <?= ($section == $link['id']) ? 'class="on" aria-current="page"' : '' ?>
              <?= html_escape($link['name']) ?>
            </a></li>
            <?php } ?>
            <li><a href="search.php">
              <span class="icon-search"></span><span class="search-text">Search</span>
            </a></li>
          </ul>
        </nav>
      </div><!-- /.container -->
    </header>
```

③
④
⑤
⑥

⑦ [

```
⑧  <footer><div class="container">&copy; Creative Folk <?= date('Y') ;?></div></footer>
</body>
<script src="js/site.js"></script>
</html>
```

Homepage

Die Startseite (`index.php`) zeigt Zusammenfassungen der sechs zuletzt auf die Website hochgeladenen Artikel.

Die Seite beginnt mit der Erfassung der Daten, die für die Erstellung der HTML-Seite benötigt werden, und speichert sie in Variablen.

1. Die strikte Typisierung ist aktiviert, um sicherzustellen, dass der richtige Datentyp verwendet wird, wenn Funktionen aufgerufen werden (siehe S. 126–127).

2. `database-connection.php` wird eingebunden, um das PDO-Objekt zu erstellen, das die Verbindung zur Datenbank verwaltet; es wird in der Variablen `$pdo` gespeichert.

3. `functions.php` wird eingebunden, da sie die Definitionen für die `pdo()`-Funktion und die Funktionen zur Formatierung der Daten, die auf der Seite angezeigt werden, enthält.

4. Die Variable `$sql` speichert das SQL, um Zusammenfassungen der neuesten Artikel zu erhalten, die der Seite hinzugefügt wurden.

5. Die SQL-Abfrage wird mit der Funktion `pdo()` ausgeführt. Sie gibt ein PDOStatement-Objekt zurück, das die Ergebnismenge repräsentiert. Die Methode `fetchAll()` des PDOStatement-Objekts ruft dann alle Zusammenfassungen als Array ab und speichert sie in der Variablen `$articles`.

6. Der SQL-Code zum Abrufen der ID und des Namens der Kategorien, die in der Navigation erscheinen, wird in `$sql` gespeichert.

7. Die Abfrage wird ausgeführt, und die Ergebnisse werden in der Variablen `$navigation` gespeichert.

8. Wenn sich ein Benutzer auf einer Kategorie- oder Artikelseite befindet, enthält `$section` die ID des Abschnitts, in dem er sich befindet. Die Homepage befindet sich nicht in einer Kategorie, daher wird hier eine leere Zeichenkette gespeichert.

9. `$title` enthält den Text für das `<title>`-Element.

10. `$description` enthält den Text, der im `<meta>`-Description-Tag angezeigt werden soll.

Den Rest der Datei unterhalb der gestrichelten Linie verwendet PHP nur, um die Daten anzuzeigen, die in den Variablen gespeichert wurden. Dies trennt den PHP-Code, der Daten abruft, von dem HTML-Code, der an den Browser zurückgesendet wird.

11. Die Datei `header.php` (wie auf der vorherigen Seite gezeigt) wird eingefügt. Sie zeigt die Daten an, die in den Schritten 6–10 abgerufen und in Variablen gespeichert wurden.

12. Eine `foreach`-Schleife arbeitet sich durch jedes Element im Array `$articles`, das in Schritt 5 erstellt wurde, und zeigt die Artikelzusammenfassungen an. Jedes Mal, wenn die Schleife läuft, werden die Daten für eine einzelne Artikelzusammenfassung in der Variablen `$article` gespeichert.

13. Es wird ein Link zu der Seite `article.php` erstellt, die jeden einzelnen Artikel auf der Website anzeigt. Der Abfragestring enthält die ID des Artikels, den er anzeigen soll.

14. Das Bild für den Artikel wird angezeigt. Wenn kein Bild angegeben wurde, wird ein Platzhalterbild angezeigt (unter Verwendung der auf S. 455 beschriebenen Technik).

15. Wenn ein Alt-Text angegeben wurde, wird er im Alt-Attribut angezeigt (andernfalls ist der Wert des Attributs leer).

16. Der Titel des Artikels wird in einem `<h2>`-Element angezeigt.

17. Die Zusammenfassung des Artikels wird angezeigt.

18. Es wird ein Link zur Datei `category.php` erstellt. Die ID der Kategorie wird in den Abfragestring eingefügt. Der Name der Kategorie wird innerhalb des Links angezeigt.

19. Es wird ein Link auf die Datei `member.php` erstellt.

Die Kennung des Mitglieds, das den Artikel verfasst hat, wird in den Abfragestring eingefügt, sodass der Link zu dessen Profilseite führt.

Der Name des Mitglieds, das den Artikel geschrieben hat, wird als Linktext verwendet.

20. Die Datei `footer.php` (wie auf der vorherigen Seite gezeigt) wird in die Seite eingefügt.

```

<?php
① declare(strict_types = 1); // Strikte Typisierung
② require 'includes/database-connection.php'; // PDO-Objekt erstellen
③ require 'includes/functions.php'; // Funktionen einfügen

[ $sql = "SELECT a.id, a.title, a.summary, a.category_id, a.member_id,
    c.name AS category,
    CONCAT(m.forename, ' ', m.surname) AS author,
    i.file AS image_file,
    i.alt AS image_alt
    FROM article AS a
    JOIN category AS c ON a.category_id = c.id
    JOIN member AS m ON a.member_id = m.id
    LEFT JOIN image AS i ON a.image_id = i.id
    WHERE a.published = 1
    ORDER BY a.id DESC
    LIMIT 6;"; // SQL für neueste Artikel
⑤ $articles = pdo($pdo, $sql)->fetchAll(); // Zusammenfassungen

⑥ $sql = "SELECT id, name FROM category WHERE navigation = 1;"; // Kategorien
⑦ $navigation = pdo($pdo, $sql)->fetchAll(); // Navigation Kategorien

⑧ $section = ''; // Aktuelle Kategorie
⑨ $title = 'Creative Folk'; // HTML <title> Inhalt
⑩ $description = 'A collective of creatives for hire'; // Meta-Description Inhalt
?>
⑪ <?php include 'includes/header.php'; ?>
    <main class="container grid" id="content">
⑫     <?php foreach ($articles as $article) { ?>
        <article class="summary">
            <a href="article.php?id=<?= $article['id'] ?>">
                ">
                <h2><?= html_escape($article['title']) ?></h2>
                <p><?= html_escape($article['summary']) ?></p>
            </a>
            <p class="credit">
                Posted in <a href="category.php?id=<?= $article['category_id'] ?>">
                    <?= html_escape($article['category']) ?></a>
                by <a href="member.php?id=<?= $article['member_id'] ?>">
                    <?= html_escape($article['author']) ?></a>
                </p>
            </article>
        <?php } ?>
    </main>
⑯ <?php include 'includes/footer.php'; ?>

```

KATEGORIENSEITE

Die Datei category.php zeigt den Namen und die Beschreibung einer einzelnen Kategorie an, gefolgt von Zusammenfassungen der Artikel in dieser Kategorie.

1. Die strikte Typisierung ist aktiviert, um sicherzustellen, dass der richtige Datentyp verwendet wird, wenn Funktionen aufgerufen werden.
2. database-connection.php und functions.php werden in die Seite eingebunden.
3. filter_input() sucht im Abfragestring nach id im Abfragestring und prüft, ob der Wert eine ganze Zahl ist. Wenn ja, enthält \$id die Zahl; wenn nicht, false. Wenn der Wert nicht in der Abfragezeichenkette enthalten war, steht hier null.
4. Wenn der Abfragestring keine gültige Ganzzahl enthielt, wird page-not-found.php eingefügt (exit verhindert die Ausführung von weiterem Code).
5. Die SQL-Abfrage, um die ID, den Namen und die Beschreibung der angegebenen Kategorie zu erhalten, wird in \$sql gespeichert.
6. Mit der Funktion pdo() wird die SQL-Abfrage ausgeführt, dann ruft die fetch()-Methode des PDOStatement-Objekts die Daten ab und speichert sie in der Variablen \$category.
7. Wenn die Kategorie nicht in der Datenbank gefunden wurde, wird page-not-found.php eingefügt, und die Seite wird angehalten.
8. Wenn die Seite noch läuft, enthält die Variable \$sql der SQL-Code, um die Zusammenfassungen der Artikel in der gewählten Kategorie abzurufen.
9. Die Abfrage wird ausgeführt, die Methode fetchAll() des PDOStatement-Objekts ruft die Zusammenfassungen als Array ab und speichert sie in der Variablen \$articles.
10. Der SQL-Code, um die ID und den Namen der Kategorien zu erhalten, die in der Navigation erscheinen, wird in \$sql gespeichert.
11. Die Abfrage wird ausgeführt, die Methode fetchAll() erhält die Daten, die sie zurückgibt, und diese werden in \$navigation gespeichert.

12. Die ID der Kategorie wird in \$section gespeichert. Mit ihr wird diese Kategorie in der Navigation hervorgehoben.

13. Der Name der Kategorie wird in \$title gespeichert. Er wird in dem <title>-Element angezeigt.
14. Die Beschreibung der Kategorie wird in \$description gespeichert und im <meta>-Description-Tag verwendet.

Da alle für die Seite erforderlichen Daten in Variablen gespeichert sind, wird mit dem Rest der Datei unterhalb der gestrichelten Linie der HTML-Code erstellt, der an den Browser zurückgesendet wird.

15. Die Datei header.php wird in die Seite eingefügt.
 16. Der Name der Kategorie wird in ein <h1>-Element geschrieben.
 17. Die Beschreibung der Kategorie wird unter dem Namen der Kategorie in einem <p>-Element angegeben.
 18. Eine foreach-Schleife arbeitet sich durch das Array, das alle Artikelzusammenfassungen in der Kategorie enthält. Dies wurde in der Variablen \$articles in Schritt 9 gespeichert. Jedes Mal, wenn die Schleife durchläuft, wird die Zusammenfassung eines anderen Artikels in der Variablen \$article gespeichert.
 19. Der Code zur Anzeige der Zusammenfassung eines Artikels ist derselbe wie der Code zur Anzeige der Zusammenfassungen der Artikel auf der Startseite (siehe vorherige Seite).
- Wenn die Kategorie noch keine Artikel enthält, wird die Variable \$articles ein leeres Array enthalten, da die fetchAll()-Methode des PDOStatement-Objekts ein leeres Array zurückgibt, wenn keine Daten mit der SQL-Abfrage übereinstimmen.
- Wenn eine foreach-Schleife versucht, mit einem leeren Array zu arbeiten, werden die Anweisungen in der Schleife nicht ausgeführt. Das bedeutet, dass die Seite keinen Undefined index-Fehler anzeigt, wenn eine Abfrage keine Daten zurückgibt.
20. Die Datei footer.php wird in die Seite eingefügt.

```

<?php

① declare(strict_types = 1); // Strikte Typisierung
② require 'includes/database-connection.php'; // PDO-Objekt erstellen
③ require 'includes/functions.php'; // functions einfügen
④ $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // id validieren
    if (!$id) { // Wenn nicht valide
        include 'page-not-found.php'; // Page not found
    }
⑤ $sql = "SELECT id, name, description FROM category WHERE id=:id;"; // SQL
⑥ $category = pdo($pdo, $sql, [$id])->fetch(); // Kategoriedaten
    if (!$category) { // Kategorie nicht gefunden
        include 'page-not-found.php'; // Page not found
    }

⑦ $sql = "SELECT a.id, a.title, a.summary, a.category_id, a.member_id,
    c.name AS category,
    CONCAT(m.forename, ' ', m.surname) AS author,
    i.file AS image_file,
    i.alt AS image_alt
    FROM article AS a
    JOIN category AS c ON a.category_id = c.id
    JOIN member AS m ON a.member_id = m.id
    LEFT JOIN image AS i ON a.image_id = i.id
    WHERE a.category_id = :id AND a.published = 1
    ORDER BY a.id DESC;"; // SQL-Statement
⑧ $articles = pdo($pdo, $sql, [$id])->fetchAll(); // Artikel

⑨ $sql = "SELECT id, name FROM category WHERE navigation = 1;"; // SQL für Kategorien
⑩ $navigation = pdo($pdo, $sql)->fetchAll(); // Navigation Kategorien
⑪ $section = $category['id']; // Aktuelle Kategorie
⑫ $title = $category['name']; // HTML <title> Inhalt
⑬ $description = $category['description']; // Meta description Inhalt
    ?>
⑭ <?php include 'includes/header.php'; ?>
    <main class="container" id="content">
        <section class="header">
            <h1><?= html_escape($category['name']) ?></h1>
            <p><?= html_escape($category['description']) ?></p>
        </section>
        <section class="grid">
            <?php foreach ($articles as $article) { ?>
                <!-- Der Code für die Artikelzusammenfassung entspricht dem auf Seite 465 -->
            <?php } ?>
        </section>
    </main>
    <?php include 'includes/footer.php'; ?>

```

ARTIKELSEITE

Mit der Datei `article.php` wird jeder einzelne Artikel auf der Website angezeigt. Der Abfragestring enthält die Kennung des Artikels, der auf der Seite angezeigt werden soll.

Wie bei den anderen Seiten werden auch in dieser Datei zunächst die Daten aus der Datenbank abgefragt und in Variablen gespeichert.

1. Die strikte Typisierung wird aktiviert, und die erforderlichen Dateien `database-connection.php` und `functions.php` werden in die Seite eingefügt.
2. `filter_input()` sucht nach `id` im Abfragestring und prüft, ob der Wert eine ganze Zahl ist.

Wenn ja, enthält `$id` die Zahl; wenn nicht, enthält es `false`.

Wenn der Wert nicht in der Abfragezeichenkette enthalten war, steht hier `null`.

3. Wenn der Abfragestring keine gültige Ganzzahl enthielt, wird `page-not-found.php` eingefügt (sie endet mit dem Befehl `exit`, der die Ausführung von weiterem Code verhindert).

4. Der SQL-Code zum Abrufen der Artikeldaten wird in `$sql` gespeichert.

5. Mit der Funktion `pdo()` wird die SQL-Abfrage ausgeführt, und mit der Methode `fetch()` werden die Artikeldaten abgerufen, die in `$article` gespeichert werden.

6. Wenn der Artikel nicht in der Datenbank gefunden wurde, wird `page-not-found.php` eingefügt und die Seite angehalten.

7. Wenn die Seite noch läuft, wird der SQL-Code zum Abrufen der Kategorien in der Navigation in `$sql` gespeichert.

8. Die Abfrage wird ausgeführt, dann ruft die `fetchAll()`-Methode die Daten ab, und sie werden in `$navigation` gespeichert.

9. Die ID der Kategorie, zu der der Artikel gehört, wird in `$section` gespeichert, damit sie in der Navigationsleiste hervorgehoben werden kann.

10. Der Titel des Artikels wird in `$title` gespeichert.

11. Die Zusammenfassung des Artikels wird in `$description` gespeichert.

Da alle für die Seite erforderlichen Daten in Variablen gespeichert sind, wird im Rest der Datei unterhalb der gestrichelten Linie der HTML-Code erstellt, der an den Browser zurückgesendet wird.

12. Die Datei `header.php` wird in die Seite eingefügt.

13. Wenn ein Bild für den Artikel hochgeladen wurde, wird sein Dateiname in das `src`-Attribut eines ``-Tags geschrieben.

Wurde kein Bild für den Artikel hochgeladen, wird stattdessen ein Platzhalterbild angezeigt.

14. Wenn für das Bild ein `alt`-Text angegeben wurde, wird dieser im `alt`-Attribut des ``-Tags angezeigt. Ist dies nicht der Fall, bleibt das Attribut leer.

15. Der Titel des Artikels wird in einem `<h1>`-Element angezeigt.

16. Das Datum, an dem der Artikel erstellt wurde, wird angezeigt.

Die Funktion `format_date()` auf S. 454 stellt sicher, dass das Datum einheitlich formatiert ist.

17. Der Inhalt des Artikels wird angezeigt.

18. Es wird ein Link auf die Datei `categpry.php` erstellt.

Die Kennung der Kategorie, in der sich der Artikel befindet, wird dem Abfragestring hinzugefügt, sodass die entsprechende Kategorieseite angezeigt wird.

19. Der Name der Kategorie wird als Linktext verwendet.

20. Es wird ein Link auf die Datei `member.php` erstellt. Die ID des Mitglieds, das den Artikel geschrieben hat, wird dem Abfragestring hinzugefügt, sodass das Profil dieses Mitglieds angezeigt wird.

21. Der Name des Mitglieds, das den Artikel geschrieben hat, wird als Linktext verwendet.

22. Die Datei `footer.php` wird in die Seite eingefügt.

```
<?php
declare(strict_types = 1); // Strikte Typisierung
① require 'includes/database-connection.php'; // PDO-Objekt
require 'includes/functions.php'; // Funktionen einfügen
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // id validieren
if (!$id) { // Wenn nicht valide
③     include 'page-not-found.php'; // Page not found
}

$sql = "SELECT a.title, a.summary, a.content, a.created, a.category_id, a.member_id,
        c.name      AS category,
        CONCAT(m.forename, ' ', m.surname) AS author,
        i.file AS image_file, i.alt  AS image_alt
④    FROM article    AS a
        JOIN category   AS c ON a.category_id = c.id
        JOIN member      AS m ON a.member_id   = m.id
        LEFT JOIN image   AS i ON a.image_id   = i.id
        WHERE a.id = :id AND a.published = 1;"; // SQL-Statement
⑤ $article = $article = pdo($pdo, $sql, [$id])->fetch(); // Artikeldaten
if (!$article) { // Wenn nicht gefunden
⑥     include 'page-not-found.php'; // Page not found
}

⑦ $sql = "SELECT id, name FROM category WHERE navigation = 1;"; // SQL für Kategorien
⑧ $navigation = pdo($pdo, $sql)->fetchAll(); // Navigationskategorien
⑨ $section    = $article['category_id']; // Aktuelle Kategorie
⑩ $title      = $article['title']; // HTML <title> Inhalt
⑪ $description = $article['summary']; // Meta description Inhalt
?>
⑫ <?php include 'includes/header.php'; ?>
    <main class="article container">
        <section class="image">
            ">
        </section>
        <section class="text">
            <h1><?= html_escape($article['title']) ?></h1>
            <div class="date"><?= format_date($article['created']) ?></div>
            <div class="content"><?= html_escape($article['content']) ?></div>
            <p class="credit">
                Posted in <a href="category.php?id=<?= $article['category_id'] ?>">
                    <?= html_escape($article['category']) ?></a>
                by <a href="member.php?id=<?= $article['member_id'] ?>">
                    <?= html_escape($article['author']) ?></a>
                </p>
            </section>
        </main>
⑯ <?php include 'includes/footer.php'; ?>
```

MITGLIEDERSEITE

Die Datei member.php zeigt die Details eines einzelnen Mitglieds und Zusammenfassungen der Artikel, die es geschrieben hat. Der Abfragestring enthält die ID des Mitglieds, das auf der Seite angezeigt werden soll.

1. Die strikte Typisierung wird aktiviert und die erforderlichen Dateien database-connection.php und functions.php werden in die Seite eingefügt.
2. filter_input() sucht nach id im Abfragestring und prüft, ob der Wert eine ganze Zahl ist.
Wenn ja, enthält \$id die Zahl; wenn nicht, false.
Wenn der Wert nicht im Abfragestring enthalten war, steht hier null.
3. Wenn der Abfragestring keine gültige Ganzzahl enthielt, wird page-not-found.php eingefügt (der Befehl exit verhindert die Ausführung von weiterem Code).
4. Der SQL-Code zum Abrufen der Mitgliedsdaten wird in \$sql gespeichert.
5. Die pdo()-Funktion führt die SQL-Abfrage aus, dann ruft die fetch()-Methode die Mitgliederdaten, die in \$member gespeichert sind, ab.
6. Wenn \$member nicht in der Datenbank gefunden wurde, wird page-not-found.php eingefügt und die Seite angehalten.
7. Wenn die Seite noch läuft, wird die SQL-Methode zum Abrufen der Zusammenfassungen der Artikel dieses Mitglieds in \$sql gespeichert.
8. Die Abfrage wird ausgeführt, die Methode fetchAll() erhält die Daten, die sie zurückgibt, und die Daten werden in \$articles gespeichert.
9. Der SQL-Code, um die Kategorien für die Navigation zu erhalten, wird in \$sql gespeichert.
10. Die Abfrage wird ausgeführt, die Methode fetchAll() - Methode erhält die Daten, die sie zurückgibt, und sie werden in \$navigation gespeichert.
11. Weil die Mitgliederseite sich nicht in einer Kategorie befindet, ist die Variable \$section leer.

12. Der Name des Mitglieds wird in \$title gespeichert, damit er im Titel der Seite angezeigt werden kann.

13. Der Name des Mitglieds, gefolgt von den Wörtern on Creative Folk, wird in \$description gespeichert, um im <meta>-Description-Tag verwendet zu werden.

Nachdem die erforderlichen Daten aus der Datenbank entnommen und in Variablen gespeichert wurden, erstellt der Rest der Datei unterhalb der gestrichelten Linie den HTML-Code, der an den Browser zurückgesendet wird.

14. Die Datei header.php wird in die Seite eingefügt.
15. Der Vor- und Nachname des Mitglieds wird in ein <h1>-Element geschrieben.
16. Das Datum des Beitritts wird mit der Funktion format_date() angezeigt, um sicherzustellen, dass das Datum einheitlich formatiert ist.
17. Wenn der Benutzer ein Profilbild hochgeladen hat, wird der Dateiname in das src-Attribut eines -Tags geschrieben. Wurde kein Bild angegeben, wird stattdessen ein Platzhalterbild angezeigt.
18. Der Vorname des Mitglieds wird im alt-Attribut des -Tags angezeigt.
19. Eine foreach-Schleife arbeitet sich durch das Array \$articles, das in Schritt 7 erstellt wurde. Es enthält Details zu den Artikeln, die dieses Mitglied geschrieben hat.
20. Der Code für die Anzeige der Zusammenfassung eines Artikels ist identisch mit dem Code für die Anzeige von Artikeln auf der Startseite, der auf S. 465 gezeigt wurde.
Wenn der Autor keine Artikel geschrieben hat, werden die Anweisungen innerhalb der Schleife nicht ausgeführt.
21. Die Datei footer.php wird in die Seite eingefügt.

```

<?php
declare(strict_types = 1); // Strikte Typisierung
① require 'includes/database-connection.php'; // PDO-Objekte erstellen
require 'includes/functions.php';
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // Funktionen einfügen
if (!$id) { // id validieren
③ include 'page-not-found.php'; // Wenn ID nicht valide
}
④ $sql = "SELECT forename, surname, joined, picture FROM member WHERE id = :id;"; // SQL
⑤ $member = pdo($pdo, $sql, [$id])->fetch(); // Mitgliederdaten
if (!$member) { // Wenn Array leer
⑥ include 'page-not-found.php'; // Page not found
}

$sql = "SELECT a.id, a.title, a.summary, a.category_id, a.member_id,
          c.name      AS category,
          CONCAT(m.forename, ' ', m.surname) AS author,
          i.file      AS image_file,
          i.alt       AS image_alt
    FROM article   AS a
    JOIN category  AS c  ON a.category_id = c.id
    JOIN member    AS m  ON a.member_id   = m.id
    LEFT JOIN image AS i  ON a.image_id   = i.id
  WHERE a.member_id = :id AND a.published = 1
        ORDER BY a.id DESC;"; // SQL
⑧ $articles = pdo($pdo, $sql, [$id])->fetchAll(); // Mitgliederartikel
⑨ $sql = "SELECT id, name FROM category WHERE navigation = 1;"; // SQL für Kategorien
⑩ $navigation = pdo($pdo, $sql)->fetchAll(); // Kategorien
⑪ $section      = ''; // Aktuelle Kategorie
⑫ $title        = $member['forename'] . ' ' . $member['surname']; // HTML <title> Inhalt
⑬ $description = $title . ' on Creative Folk'; // Meta-Description
?>

⑭ <?php include 'includes/header.php'; ?>
<main class="container" id="content">
  <section class="header">
    <h1><?= html_escape($member['forename']) . ' ' . $member['surname'] ?></h1>
    <p class="member"><b>Member since:</b> <?= format_date($member['joined']) ?></p>
    " class="profile"><br>
  </section>
  <section class="grid">
    <?php foreach ($articles as $article) { ?>
      <!-- Code für die Artikelzusammenfassungen ist derselbe wie auf S. 465 -->
      <?php } ?>
    </section>
  </main>
⑮ <?php include 'includes/footer.php'; ?>

```

EINE SUCHFUNKTION ERSTELLEN

Die Suchseite zeigt, wie man mit SQL nach Daten in der Datenbank sucht und wie man, wenn eine Datenbankabfrage viele Zeilen zurückgeben kann, die Ergebnisse auf mehreren Seiten anzeigt.

Wenn Besucher einen Begriff in das Suchfeld eingeben und das Formular abschicken, sucht die Datenbank in den Spalten `title`, `summary` und `description` der Tabelle `article` nach diesem Begriff. Wenn der Suchbegriff in diesen Spalten gefunden wird, wird eine Zusammenfassung des Artikels zu einer Ergebnismenge hinzugefügt, damit die Besucher sehen können, welche Artikel der Anfrage entsprechen.

Auf der Suchseite werden zwei SQL-Abfragen ausgeführt:

- Die eine zählt die Gesamtzahl der übereinstimmenden Ergebnisse.
- Mit der anderen werden die Details dieser Artikel zusammengefasst.

Unten sehen Sie die SQL-Abfrage, die die Anzahl der Artikel zählt, die den Suchbegriff enthalten.

Der Suchbegriff wird in der Abfrage dreimal wiederholt, zuerst in der Spalte `title`, dann in der Spalte `summary` und schließlich in der Spalte `content`.

Da ein Platzhalter in einer SQL-Abfrage nicht wiederverwendet werden kann, wird für die Suche in jeder Spalte ein anderer Platzhaltername verwendet (`:term1`, `:term2` und `:term3`).

```
SELECT COUNT(title)
  FROM article
 WHERE title  LIKE :term1
   OR summary LIKE :term2
   OR content LIKE :term3
  AND published = 1;
```

Es werden drei Suchergebnisse pro Seite angezeigt, um zu demonstrieren, wie Sie

- eine Teilmenge der Ergebnisse pro Seite anzeigen können, wenn die Datenbank viele übereinstimmende Ergebnisse findet.
- Links unter den Ergebnissen einfügen, über die die nächste (oder vorherige) Gruppe von Übereinstimmungen angefordert werden kann.

Dies wird als **Paginierung** bezeichnet, da die Ergebnisse auf mehrere Seiten aufgeteilt werden. Die Links zu den einzelnen Seiten, auf denen die Suchergebnisse angezeigt werden, benötigen drei Name/Wert-Paare im Abfragestring, damit die Suchseite weiß, welche Artikel sie aus der Datenbank abrufen und anzeigen soll:

- `term` enthält den Suchbegriff.
- `show` gibt an, wie viele Ergebnisse pro Seite angezeigt werden sollen.
- `from` gibt an, wie viele Treffer angezeigt wurden.

Der `Show`-Wert wird mit der SQL-LIMIT-Klausel verwendet, damit die Datenbank die richtige Anzahl von Artikeln für die Anzeige auf der Seite liefert.

Der `from`-Wert wird mit der SQL-OFFSET-Klausel verwendet, um der Datenbank mitzuteilen, dass sie erst dann mit dem Hinzufügen von Ergebnissen zur Ergebnismenge beginnen soll, wenn sie die angegebene Anzahl von Übereinstimmungen gefunden hat.

```
LIMIT :show
OFFSET :from
```

**Travel Guide**

Book design for series of travel guides
POSTED IN PRINT BY IVY STONE

Golden Brown

Photograph for interior design book
POSTED IN PHOTOGRAPHY BY EMIKO ITO

Polite Society Posters

Poster designs for a fashion label
POSTED IN PRINT BY IVY STONE

1 2 3 4

© CreativeFolk 2021

search.php?term=design

search.php?term=design&show=3&from=3

search.php?term=design&show=3&from=6

search.php?term=design&show=3&from=9

VON	ZEIGE	SEITE
(0 ÷ 3)	+ 1	= 1
(3 ÷ 3)	+ 1	= 2
(6 ÷ 3)	+ 1	= 3
(9 ÷ 3)	+ 1	= 4

Um die Paginierungslinks zu erstellen, benötigt die Suchseite drei Daten, die jeweils in einer Variablen gespeichert werden:

- \$count ist die Anzahl der Ergebnisse, die der Suchanfrage entsprechen.
- \$show ist die Anzahl der Ergebnisse, die pro Seite angezeigt werden sollen.
- \$from ist die Anzahl der Ergebnisse, die vor dem Hinzufügen von Übereinstimmungen zur Ergebnismenge übersprungen werden.

Um die Anzahl der Seiten zu berechnen, die für die Anzeige der Ergebnisse benötigt werden, dividieren Sie \$count (die Anzahl der Treffer) durch \$show (die Anzahl der pro Seite angezeigten Ergebnisse). Oben gibt es 10 Treffer, und 3 Treffer werden pro Seite angezeigt. $10 \div 3 = 3,3333$, und diese Zahl wird mit der PHP-Funktion ceil() aufgerundet, um die Anzahl der Seiten zu ermitteln, die für die Anzeige der Ergebnisse benötigt werden.

```
$total_pages = ceil($count / $show);
```

Um die aktuelle Seite zu ermitteln, dividieren Sie \$from (die Anzahl der zu überspringenden Ergebnisse) durch \$show (die Anzahl der Ergebnisse pro Seite) und addieren dann 1. Die Berechnungen zur Ermittlung der aktuellen Seite sind oben rechts dargestellt.

```
$current_page = ceil($from / $show) + 1;
```

Mit einer for-Schleife werden die Paginierungslinks erstellt. Sie setzt einen Zähler auf 1 und prüft, ob dieser Zähler kleiner ist als die Gesamtzahl der Seiten. Ist dies der Fall, wird ein Link zur Seite hinzugefügt und der Zähler erhöht. Die Schleife läuft so lange, bis der Zähler die Gesamtzahl der Seiten erreicht hat, die für die Anzeige der Ergebnisse erforderlich sind.

```
for ($i = 1; $i <= $total_pages; $i++) {
    // Einen weiteren Link anzeigen
}
```

SUCHSEITE

Wird ein Suchbegriff in das Formular oben auf der Seite `search.php` eingegeben und das Formular abgeschickt, wird der Suchbegriff an dieselbe Seite zurückgeschickt, die dann passende Artikel findet und die Ergebnisse anzeigt.

1. Strikte Typisierung ist aktiviert, und `functions.php` und `database-connection.php` werden in die Seite eingefügt.

2. `filter_input()` ruft drei Werte aus dem Abfragestring ab und speichert sie in Variablen:

- `$term` enthält den Suchbegriff.
- `$show` gibt die Anzahl der Ergebnisse an, die pro Seite angezeigt werden sollen (wenn kein Wert angegeben wird, wird die Standardzahl 3 verwendet).
- `$from` liefert die Anzahl der zu überspringenden Ergebnisse (wenn kein Wert angegeben wird, wird standardmäßig 0 verwendet).

3. Zwei Variablen werden initialisiert, da sie für die Erstellung der HTML-Seite benötigt werden, ihnen werden aber nur Werte zugewiesen, wenn es einen Suchbegriff gibt (`$count` hat den Wert 0, und `$articles` ist ein leeres Array).

4. Eine `if`-Anweisung prüft, ob `$term` einen Suchbegriff enthält; dies deshalb, weil der nächste Teil der Seite, der versucht, Übereinstimmungen in der Datenbank zu finden, nur ausgeführt werden soll, wenn ein Suchbegriff eingegeben wurde.

5. Das Array `$arguments` speichert die Namen der drei Platzhalter in der SQL-Abfrage und die Werte, durch die sie ersetzt werden sollen. Jeder Platzhalter wird durch denselben Suchbegriff ersetzt, da SQL-Abfragen denselben Platzhalternamen nicht wieder verwenden können.

Das Platzhaltersymbol `%` wird vor und nach dem Suchbegriff eingefügt, damit die SQL-Abfrage auch dann Übereinstimmungen finden kann, wenn Zeichen auf beiden Seiten des Suchbegriffs vorkommen.

6. Die erste SQL-Abfrage zählt, wie viele Artikel in der Tabelle `article` den Suchbegriff entweder in den Spalten `title`, `summary` oder `content` enthalten.

7. Mit der Funktion `pdo()` wird die SQL-Abfrage ausgeführt (unter Verwendung des angegebenen Suchbegriffs).

Dann wird mit der Methode `fetchColumn()` des `PDOStatement`-Objekts die Anzahl der Artikel ermittelt, die mit dem Suchbegriff übereinstimmen. Wie der Name sagt, ruft die `fetchColumn()`-Methode den Wert aus einer einzelnen Spalte der Ergebnismenge ab. Der zurückgegebene Wert wird in `$count` gespeichert.

8. Eine `if`-Anweisung prüft, ob es Übereinstimmungen gab. Wenn ja, werden die Artikelzusammenfassungen abgerufen. Wenn nicht, ist der Versuch, sie abzurufen, sinnlos.

9. Zwei weitere Elemente werden dem Array `$arguments` hinzugefügt, damit sie in der zweiten SQL-Abfrage verwendet werden können:

- `show` ist die Anzahl der Ergebnisse, die pro Seite angezeigt werden sollen.
- `from` ist die Anzahl der Ergebnisse, die übersprungen werden sollen (beide wurden in Schritt 2 in Variablen gespeichert).

10. Die zweite SQL-Abfrage erhält die Zusammenfassungen der passenden Artikel, die auf dieser Seite angezeigt werden; sie wird in `$sql` gespeichert.

11. Die `WHERE`-Klausel findet die Artikel, die den Suchbegriff entweder in den Spalten `title`, `summary` oder `content` der Tabelle `article` enthalten.

12. Die `ORDER BY`-Klausel ordnet die Ergebnisse nach Artikel-ID in absteigender Reihenfolge an, sodass die neuesten Artikel zuerst angezeigt werden.

13. Die `LIMIT`-Klausel schränkt die Anzahl der Ergebnisse, die der Ergebnismenge hinzugefügt werden, auf die Anzahl der Suchergebnisse ein, die pro Seite angezeigt werden sollen.

14. Die `OFFSET`-Klausel steuert, wie viele Treffer übersprungen werden, bevor die Daten zur Ergebnismenge hinzugefügt werden.

15. Die SQL-Anweisung wird unter Verwendung der Werte im aktualisierten Array `$arguments` ausgeführt. Anschließend wird die Methode `fetchAll()` des `PDOStatement`-Objekts aufgerufen, um alle übereinstimmenden Zusammenfassungen zu erhalten. Sie werden in `$articles` gespeichert.

```
<?php  
① declare(strict_types = 1); // Strikte Typisierung  
require 'includes/database-connection.php'; // PDO-Objekt erstellen  
require 'includes/functions.php'; // Funktionen einfügen  
  
② $term  = filter_input(INPUT_GET, 'term'); // Suchbegriff  
③ $show  = filter_input(INPUT_GET, 'show', FILTER_VALIDATE_INT) ?? 3; // Limit  
④ $from  = filter_input(INPUT_GET, 'from', FILTER_VALIDATE_INT) ?? 0; // Offset  
⑤ $count = 0; // Zähler auf 0  
$articles = [];  
  
⑥ if ($term) { // Wenn Suchbegriff  
    $arguments['term1'] = '%$term%'; // Suchbegriff in Array speichern  
    $arguments['term2'] = '%$term%'; // Dreimal als Platzhalter  
    $arguments['term3'] = '%$term%'; // Kann in SQL nicht wiederholt werden  
  
    $sql = "SELECT COUNT(title) FROM article  
          WHERE title LIKE :term1  
            OR summary LIKE :term2  
            OR content LIKE :term3  
            AND published = 1;"; // Wie viele Artikel entsprechen Suchbegriff  
    $count = pdo($pdo, $sql, $arguments)->fetchColumn();  
    if ($count > 0) {  
        $arguments['show'] = $show;  
        $arguments['from'] = $from;  
        $sql = "SELECT a.id, a.title, a.summary, a.category_id, a.member_id,  
              c.name      AS category,  
              CONCAT(m.forename, ' ', m.surname) AS author,  
              i.file      AS image_file,  
              i.alt       AS image_alt  
        FROM article   AS a  
        JOIN category  AS c  ON a.category_id = c.id  
        JOIN member    AS m  ON a.member_id   = m.id  
        LEFT JOIN image AS i  ON a.image_id   = i.id  
        WHERE a.title  LIKE :term1  
          OR a.summary LIKE :term2  
          OR a.content LIKE :term3  
          AND a.published = 1  
        ORDER BY a.id DESC  
        LIMIT :show  
        OFFSET :from;"  
        $articles = pdo($pdo, $sql, $arguments)->fetchAll();  
    }  
}
```

SUCHSEITE (FORTSETZUNG)

Als Nächstes werden die Werte berechnet, die für die Erstellung der Pagenumberlinks benötigt werden.

1. Wenn die Zahl in \$count größer ist als \$show, müssen die Werte für die Pagenumberlinks berechnet werden.
2. Die Gesamtzahl der Seiten, die für die Anzeige der Ergebnisse benötigt werden (gespeichert in \$total_pages), wird berechnet durch:
 - Division von \$count durch \$show
 - Rundung mit PHPs ceil() -Funktion
3. Die aktuelle Seite (gespeichert in \$current_page) wird berechnet durch:
 - Dividieren des Werts in \$show durch den Wert in \$from
 - Aufrunden mithilfe von PHPs ceil() -Funktion
 - Hinzufügen von 1 zu der Zahl
4. Der SQL-Code, um die Kategorien zu erhalten, damit sie in der Navigationsleiste angezeigt werden können, wird in \$sql gespeichert.
5. Die Abfrage wird ausgeführt, die Methode fetchAll() ruft die Kategorien ab und speichert sie in \$navigation.
6. Da sich die Suchseite nicht in einer Kategorie befindet, speichert die Variable \$section einen leeren String.
7. Der Titel der Seite wird in \$title gespeichert. Er besteht aus dem Text Search results for und dem Suchbegriff (dieser wird in header.php maskiert).
8. Der Text der Meta-Beschreibung wird in der Variablen \$description gespeichert.
- Der Rest der Datei erstellt die HTML-Seite, die an den Browser zurückgesandt wird.
9. Das Suchformular sendet die Daten an diese Seite zurück.
10. Wenn der Benutzer einen Suchbegriff eingegeben hat, wird dieser bereinigt und in der Sucheingabe angezeigt.
11. Wenn es einen Wert in \$term gibt, wird die Anzahl der passenden Artikel angezeigt.

12. Die Zusammenfassungen der Artikel werden in einer foreach-Schleife angezeigt, wie bei den vorherigen Beispielen (siehe S. 465).
13. Eine if-Anweisung prüft, ob \$count eine größere Zahl als \$show enthält. Ist dies der Fall, werden Pagenumberlinks angezeigt.
14. Ein <nav>- und ein -Element werden hinzugefügt, um die Pagenumberlinks aufzunehmen. Jeder Link wird in einem -Element untergebracht.
15. Mit einer for-Schleife werden nun die Pagenumberlinks erstellt.

Die Klammern enthalten drei Ausdrücke:

 - \$i = 1 erzeugt den Zähler \$i und setzt ihn auf 1.
 - \$i <= \$total_pages ist die Bedingung, die prüft, ob der Code in der Schleife ausgeführt werden soll. Ist der Zähler kleiner als die Gesamtzahl der Seiten, die für die Anzeige der Suchergebnisse benötigt werden, wird der folgende Codeblock ausgeführt.
 - Mit \$i++ wird der Zähler bei jedem Durchlauf um 1 erhöht.
16. Innerhalb der Schleife wird für jede Seite ein Link erstellt. Das href-Attribut verwendet einen Abfragestring mit den drei Werten, die search.php mitteilen, welche Ergebnisse angezeigt werden sollen:
 - term ist der Suchbegriff.
 - show ist die Anzahl der Ergebnisse, die pro Seite angezeigt werden sollen.
 - from ist die Anzahl der Ergebnisse, die übersprungen werden sollen. Zum Beispiel: Für Seite eins ist \$i 1, also (1 - 1) * 3, null Ergebnisse werden übersprungen; Seite zwei: \$i ist 2, (2 - 1) * 3, drei Ergebnisse werden übersprungen.
17. Wenn der Wert im Zähler mit dem Wert in \$current_page übereinstimmt, sollte der Link anzeigen, dass dies die aktuelle Ergebnisseite ist. Dazu wird der Wert active zum class-Attribut hinzugefügt und ein aria-current-Attribut mit einem Wert von true.
18. Innerhalb des Links wird der Zähler als Linktext verwendet, um die Seitenanzahl anzuzeigen. Die Schleife läuft so lange, bis der Wert im Zähler gleich dem Wert in \$total_pages ist.

```
① if ($count > $show) {  
②     $total_pages = ceil($count / $show);           // berechne Gesamtseiten  
③     $current_page = ceil($from / $show) + 1;        // berechne aktuelle Seite  
}  
④ $sql = "SELECT id, name FROM category WHERE navigation = 1;" // SQL für Kategorien  
⑤ $navigation = pdo($pdo, $sql)->fetchAll();                  // Navigationskategorien abfragen  
  
⑥ $section      = '';                                         // aktuelle Kategorie  
⑦ $title        = 'Search results for ' . $term;             // HTML <title>  
⑧ $description = $title . ' on Creative Folk';               // Meta-Beschreibung  
?  
<?php include 'includes/header.php'; ?>  
<main class="container" id="content">  
    <section class="header">  
        <form action="search.php" method="get" class="form-search">  
            <label for="search"><span>Search for: </span></label>  
            <input type="text" name="term" value=<?= html_escape($term) ?>  
                id="search" placeholder="Enter search term"  
            /><input type="submit" value="Search" class="btn" />  
        </form>  
        <?php if ($term) { ?><p><b>Matches found:</b> <?= $count ?></p><?php } ?>  
    </section>  
  
    <section class="grid">  
        <?php foreach ($articles as $article) { ?>  
        <!-- Code zur Anzeige der Artikelzusammenfassungen entspricht S. 465 --&gt;<br/>        <?php } ?>  
    </section>  
  
    <?php if ($count > $show) { ?>  
    <nav class="pagination" role="navigation" aria-label="Pagination navigation">  
        <ul>  
            <?php for ($i = 1; $i <= $total_pages; $i++) { ?>  
                <li>  
                    <a href=?term=<?= $term ?>&show=<?= $show ?>&from=<?= ((($i - 1) * $show) ?>"  
                        class="btn <?= ($i == $current_page) ? 'active' aria-current="true' : '' ?>"  
                        <?= $i ?>  
                    </a>  
                </li>  
            <?php } ?>  
        </ul>  
    </nav>  
    <?php } ?>  
    </main>  
<?php include 'includes/footer.php'; ?>
```

DATEN IN EIN OBJEKT ÜBERTRAGEN

PDO kann jede Datenzeile in einer Ergebnismenge als Objekt statt als Arrays repräsentieren (und die Methoden dieses Objekts können mit den zurückgegebenen Daten arbeiten). Der Fetch-Modus von PDO legt fest, wie die Daten dargestellt werden sollen.

Fetch-Modi steuern, wie ein PDOStatement-Objekt jede Datenzeile in der Ergebnismenge zurückgibt:

- Assoziatives Array – jeder Spaltenname aus der Ergebnismenge wird als Schlüssel des Arrays verwendet.
- Objekt – jeder Spaltenname aus der Ergebnismenge wird als Objekteigenschaft verwendet.

Die Datei database-connection.php auf S. 439 verwendet das Array \$options, um den Standard-Fetch-Modus so einzustellen, dass jede Datenzeile als assoziatives Array zurückgegeben wird.

Um den Standard-Fetch-Modus so einzustellen, dass jede Datenzeile als Objekt statt als Array zurückgegeben wird, verwenden Sie den Wert

PDO::FETCH_OBJ.

```
ARRAY    . PDO::ATTR_DEFAULT_FETCH_MODE =>
OBJEKT   . PDO::FETCH_ASSOC;
          PDO::ATTR_DEFAULT_FETCH_MODE =>
          PDO::FETCH_OBJ;
```

Jedes PDOStatement-Objekt verfügt auch über die Methode setFetchMode(), mit der der Abrufmodus für das jeweilige PDOStatement-Objekt festgelegt werden kann.

Damit wird die Standard-Fetch-Methode überschrieben.

setFetchMode() wird nach der Methode execute() aufgerufen. Der einzige Parameter ist der zu verwendende Abrufmodus. Im Folgenden soll jede Datenzeile aus der Ergebnismenge als Objekt zurückgegeben werden.

```
$statement->setFetchMode(PDO::FETCH_OBJ);
```

PDOStatement-OBJEKT ABRUFMODUS JEDE ZEILE ALS OBJEKT

Der Abrufmodus kann auch als Argument der Methoden fetch() und fetchAll() angegeben werden. Wenn mit fetchAll() mehrere Datenzeilen aus der Ergebnismenge abgerufen werden, wird jedes Objekt in einem separaten Element eines indizierten Arrays gespeichert.

Das Objekt wird mit der sogenannten **Standardklasse** erstellt; dies ist ein leeres Objekt (ohne Eigenschaften oder Methoden). Der Name der Klasse ist stdClass. Das Beispiel auf S. 480 zeigt, wie die Daten zu einem Objekt hinzugefügt werden können, das mit einer vorhandenen Klasse erstellt wurde.

```
$statement->fetch(PDO::FETCH_OBJ);
```

PDOStatement-OBJEKT DATEN ABRUFEN JEDE ZEILE ALS OBJEKT

DEN FETCH-MODUS ZUM ABRUFEN EINES OBJEKTS EINSTELLEN

PHP section_c/c12/examples/fetching-data-as-objects.php

```
<?php
① require '../cms/includes/database-connection.php';
② require '../cms/includes/functions.php';
③ $sql = "SELECT id, forename, surname
④           FROM member"; // SQL
⑤ $statement = $pdo->query($sql); // Ausführen
⑥ $statement->setFetchMode(PDO::FETCH_OBJ); // FetchMode
⑦ $members = $statement->fetchAll(); // Daten holen
?>
<!DOCTYPE html>
<html> ...
<body>
⑥   <?php foreach ($members as $member) { ?>
      <p>
        <?= html_escape($member->forename) ?>
        <?= html_escape($member->surname) ?>
      </p>
    <?php } ?>
</body>
</html>
```

ERGEBNIS

Ivy Stone

Luke Wood

Emiko Ito

Probieren Sie es: Fordern Sie in Schritt 2 auch die E-Mail-Adresse des Mitglieds an und zeigen Sie diese in Schritt 7 an.

Probieren Sie es: Ersetzen Sie Schritt 7 durch <?php var_dump(\$Mitglied) ?>, um das für jede Datenzeile erstellte Objekt zu sehen.

In diesem Beispiel wird jede Zeile der Ergebnismenge durch eine Objekteigenschaft dargestellt.

1. database-connection.php und functions.php werden eingefügt.

2. Die Abfrage wird in \$sql gespeichert.

3. Die Methode query() des PDO-Objekts führt die Abfrage aus. Sie gibt ein PDOStatement-Objekt zurück, das die Abfrage und die erzeugte Ergebnismenge repräsentiert.

4. Die Methode setFetchMode() des PDOStatement-Objekts wird aufgerufen. Das Argument PDO::FETCH_OBJ gibt an, dass jede Zeile der Ergebnismenge als Objekt zurückgegeben werden soll.

5. Die fetchAll()-Methode des PDOStatement-Objekts ruft jede Datenzeile aus der Ergebnismenge ab.

Sie gibt ein indiziertes Array zurück, und der Wert jedes Elements in diesem Array ist ein Objekt, das eine Zeile der Daten repräsentiert.

6. Mit einer foreach-Schleife wird jedes Element des Arrays durchlaufen.

7. Der Name des Mitglieds wird anhand der Objekteigenschaften angezeigt, das den Vor- und Nachnamen enthält.

DATEN MIT HILFE EINER KLASSE IN EIN OBJEKT ÜBERTRAGEN

PDO kann jede Zeile der Ergebnismenge als Objekt zurückgeben, das mit einer benutzerdefinierten Klasse erstellt wurde. Die mit dieser Klasse erstellten Objekte erhalten automatisch alle Methoden der Klassendefinition.

Um Daten aus jeder Zeile einer Ergebnismenge zu einem Objekt hinzuzufügen, das mit einer bestehenden Klassendefinition erstellt wurde, verwenden Sie die Methode `setFetchMode()` des `PDOStatement`-Objekts mit zwei Parametern:

- Abrufmodus `PDO::FETCH_CLASS`
- Name der zu verwendenden Klasse

Der Klassenname wird in Anführungszeichen gesetzt, und die Klassendefinition muss in die Seite eingefügt werden, bevor `setFetchMode()` aufgerufen wird.

Auf der rechten Seite sehen Sie eine Klassendefinition, die zum Erstellen von Objekten verwendet wird, die die Mitglieder der Site repräsentieren. Die Klasse befindet sich in der Datei `Member.php`, die im Ordner `classes` gespeichert ist.

Die Eigenschaften dieses Objekts entsprechen den Namen von zwei Spalten in der `member`-Tabelle der Datenbank.

- Wenn ein Spaltenname der Ergebnismenge mit einem Eigenschaftsnamen in der Klasse übereinstimmt, wird der Wert dieser Eigenschaft zugewiesen.
- Wenn die Ergebnismenge einen Spaltennamen hat, der keine Eigenschaft in der Klasse ist, wird dieser Spaltenname als zusätzliche Eigenschaft des Objekts hinzugefügt.

Jedes Objekt, das PDO mit dieser Klasse erstellt, verfügt auch über die Methoden, die in der Klassendefinition enthalten sind.

Wenn `fetchAll()` verwendet wird, um mehrere Datenzeilen aus der Ergebnismenge abzurufen, wird jedes Objekt in einem anderen Element in einem indizierten Array gespeichert.

```
$statement->setFetchMode(PDO::FETCH_CLASS, 'Member');
```

PDOStatement-
OBJEKT

SETFETCH
MODE

DATEN IN VOR-
HANDENE KLASSE
EINFÜGEN

KLASSENNAME

Wenn Objekte aus benannten Klassen erstellt werden, werden den Objekteigenschaften Werte zugewiesen, bevor die Methode `__construct()` der Klasse (siehe S. 160) aufgerufen wird. Dies kann zu unerwarteten Ergebnissen führen.

EIN OBJEKT MIT EINER VORHANDENEN KLASSE ERSTELLEN

PHP

section_c/c12/examples/classes/Member.php

```
① <?php
  class Member
  {
    public $forename;
    public $surname;
    public function getFullName(): string
    {
      return $this->forename . ' ' . $this->surname;
    }
  }
```

PHP

section_c/c12/examples/fetching-data-into-class.php

```
② <?php
  require '../cms/includes/database-connection.php';
  require '../cms/includes/functions.php';
  require 'classes/Member.php';
  $sql = "SELECT forename, surname
          FROM member
          WHERE id = 1;";
  $statement = $pdo->query($sql);
  $statement->setFetchMode(PDO::FETCH_CLASS, 'Member');
  $member = $statement->fetch();
?>
<!DOCTYPE html>
<html> ...
  ⑦   <p><?= html_escape($member->getFullName()) ?></p> ...
</html>
```

ERGEBNIS

Ivy Stone

1. Die Klasse Member wird mit zwei Eigenschaften und einer Methode definiert.
2. database-connection.php, functions.php und member.php werden eingefügt.
3. Die Abfrage wird in \$sql gespeichert.
4. Die Methode query() des PDO-Objekts führt die Abfrage aus und erstellt ein PDOStatement-Objekt, um die Ergebnismenge darzustellen.
5. Die setFetchMode()-Methode des PDOStatement-Objekts wird aufgerufen:
 - PDO::FETCH_CLASS weist PDO an, die Daten zu einem Objekt hinzuzufügen, das mit einer bestehenden Klasse erstellt wurde.
 - Member ist der Name der Klasse, die zur Erstellung des Objekts verwendet wird.
6. Die fetch()-Methode des PDOStatement-Objekts ruft die Datenzeile aus der Ergebnismenge ab. Das zurückgegebene Objekt wird in der Variablen \$member gespeichert.
7. Die Methode getFullName() des Objekts wird aufgerufen, um den vollständigen Namen des Mitglieds zu erhalten.

Probieren Sie es: Ersetzen Sie Schritt 6 durch <?php var_dump(\$member) ?>, um das Objekt zu sehen, das für dieses Mitglied erstellt wird.

ZUSAMMENFASSUNG

DATEN AUS DER DATENBANK ABRUFEN

- Ein PDO-Objekt verwaltet die Verbindung zur Datenbank.
- Ein PDOStatement-Objekt stellt eine SQL-Anweisung und die von ihr erzeugte Ergebnismenge dar. Es kann jede Datenzeile als assoziatives Array oder Objekt zurückgeben.
- Wenn eine Ergebnismenge mehr als eine Zeile enthält, kann jede Zeile in einem Element eines indizierten Arrays gespeichert werden.
- Abfrage-Strings können verwendet werden, um anzugeben, welche Daten eine Seite aus der Datenbank abrufen soll.
- Die SQL-Anweisung kann Platzhalter für Werte verwenden, die sich bei jeder Anforderung der Seite ändern können.
- Stellen Sie sicher, dass alle Datenbankdaten, die von den Besuchern einer Website erstellt werden, vor der Anzeige auf einer Seite maskiert werden.

13

DATEN IN DER
DATENBANK
AKTUALISIEREN

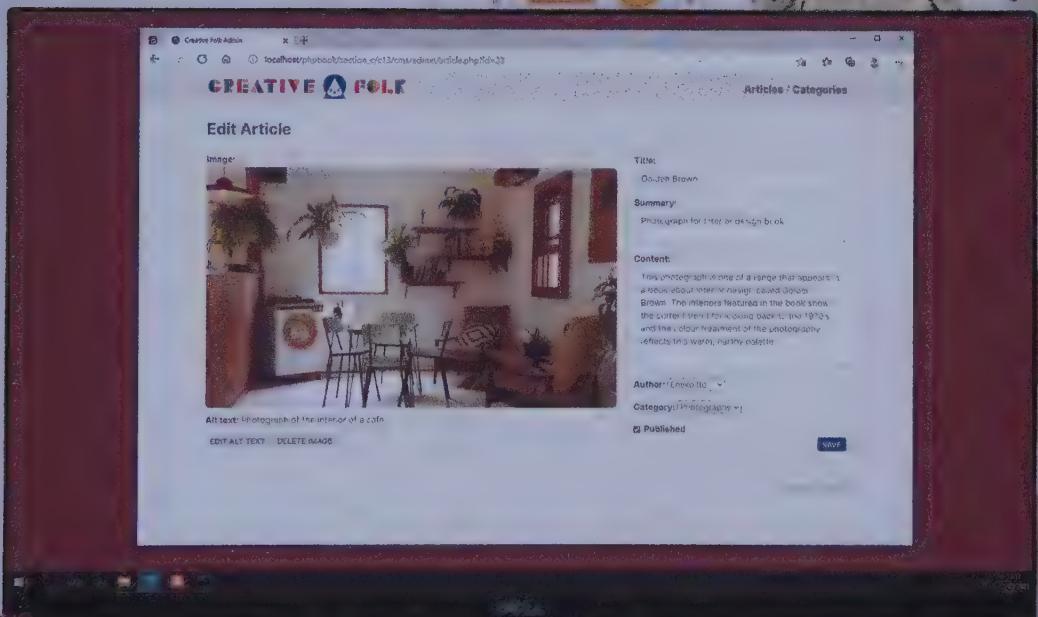
Eine Website kann Werkzeuge bereitstellen, mit denen neue Daten in die Datenbank eingegeben und bereits gespeicherte Daten aktualisiert oder gelöscht werden können.

Zu diesem Zweck müssen die PHP-Seiten folgende Aufgaben erfüllen:

- 1. Daten abrufen:** In Kapitel 6 wurde gezeigt, wie Sie Daten aus Formularen und URLs abrufen.
- 2. Daten validieren:** In Kapitel 6 wurde auch gezeigt, wie Sie prüft, ob die erforderlichen Daten eingegeben wurden und ob sie in einem gültigen Format vorliegen (und wie Sie bei Fehlern Meldungen an die Benutzer ausgeben).
- 3. Datenbank aktualisieren:** In Kapitel 11 wurden SQL-Anweisungen zum Erstellen, Aktualisieren oder Löschen von Daten in der Datenbank vorgestellt. In Kapitel 12 wurde gezeigt, wie SQL-Anweisungen mit PDO ausgeführt werden können.
- 4. Rückmeldung geben:** In einer Meldung wird den Nutzern mitgeteilt, ob sie erfolgreich waren oder nicht.

Da Sie bereits gelernt haben, wie die meisten dieser Aufgaben ausgeführt werden; konzentriert sich dieses Kapitel darauf, wie Sie steuern können, wann die einzelnen Aufgaben ausgeführt werden sollen. Die Reihenfolge, in der die Anweisungen ausgeführt werden, wird als **Kontrollfluss** bezeichnet. In diesem Kapitel wird dem PHP-Interpreter durch eine Reihe von `if`-Anweisungen mitgeteilt, wann die einzelnen Aufgaben ausgeführt werden sollen. Wenn beispielsweise die eingegebenen Daten ungültig sind, ergibt es keinen Sinn, die SQL-Anweisungen zur Aktualisierung der Datenbank zu erstellen oder auszuführen. Ebenso müssen Sie nur dann eine Erfolgsmeldung anzeigen, wenn die Änderungen an der Datenbank erfolgreich waren.

Sie lernen auch, wie Sie eine Reihe von zusammenhängenden SQL-Anweisungen mithilfe von so genannten Transaktionen ausführen und wie die Änderungen nur gespeichert werden, wenn alle SQL-Anweisungen erfolgreich sind (wenn eine von ihnen fehlschlägt, wird keine der Änderungen in der Datenbank gespeichert).



DATEN IN EINE TABELLE EINFÜGEN

Um eine neue Zeile zu einer Datenbanktabelle hinzuzufügen, verwenden Sie den SQL-Befehl `INSERT`. Ein `INSERT`-Befehl kann immer nur Daten zu einer einzigen Tabelle gleichzeitig hinzufügen.

1. Die folgende SQL-Anweisung fügt der Tabelle `category` eine Kategorie hinzu. Sie enthält Parameter für die Spalten `name`, `description` und `navigation`. (Der Wert für die Spalte `id` wird von der Datenbank generiert.)

2. Die Daten, die von den einzelnen Spalten verwendet werden, werden in einem assoziativen Array mit genau einem Element für jeden Parameter in der SQL-Anweisung bereitgestellt. Das Array darf keine zusätzlichen Elemente enthalten, da dies einen Fehler verursachen würde.

3. Die `prepare()`-Methode des PDO-Objekts benötigt eine SQL-Anweisung als Argument, damit sie ein `PDOStatement`-Objekt erstellen kann. Anschließend verwendet die Methode `execute()` des `PDOStatement`-Objekts die Werte im Array, um die SQL-Anweisung auszuführen.

```
① [ $sql = "INSERT INTO category (name, description, navigation)  
      VALUES (:name, :description, :navigation);";  
  
② [ $category = ['name']          = 'News';  
③ [ $category = ['description'] = 'News about Creative Folk';  
    $category = ['navigation']   = 1;  
  
④ [ $statement = $pdo->prepare($sql);  
⑤ [ $statement->execute($category);
```

Auf der rechten Seite wurde die neue Zeile hervorgehoben. Durch die automatische Inkrementierung erhält die Spalte `id` den Wert 5. In der Beispiel-Website würde das PDO-Objekt bei einem Problem eine Ausnahme auslösen, die von der Standardfunktion zur Behandlung von Ausnahmen abgefangen würde.

Kategorie			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	News about Creative Folk	1

DATEN IN EINER TABELLE AKTUALISIEREN

Um bestehende Zeilen in einer Datenbanktabelle zu aktualisieren, verwenden Sie den SQL-Befehl UPDATE. Ein UPDATE-Befehl kann mehrere Tabellen über einen JOIN aktualisieren.

1. Mit dieser SQL-Anweisung wird eine bestehende Kategorie aktualisiert. Die ersten drei Parameter enthalten Werte, die in den Spalten name, description und navigation zu verwenden sind. Die WHERE-Klausel verwendet einen Parameter, um die ID einer zu aktualisierenden Zeile anzugeben.

2. Die zum Ersetzen der Parameter verwendeten Daten werden in einem Array bereitgestellt, das für jeden Parameter der SQL-Anweisung genau ein Element enthält. Es darf keine zusätzlichen Elemente enthalten, da dies einen Fehler verursachen würde.

3. Die Anweisung wird wie eine Abfrage mit Parametern ausgeführt. Unten sehen Sie, dass die benutzerdefinierte Funktion pdo() (auf S. 456 erläutert) zur Ausführung der SQL-Anweisung verwendet wird. Dieser Ansatz wird auch im weiteren Verlauf des Kapitels verwendet.

```
① $sql = "UPDATE category
          SET name      = :name,
              description = :description,
              navigation = :navigation
          WHERE id = :id;";

② $category = ['id']      = 5;
$category = ['name']     = 'News';
$category = ['description'] = 'Updates from Creative Folk';
$category = ['navigation'] = 0;

③ pdo($pdo, $sql, $category);
```

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1
5	News	Updates from Creative Folk	0

Auf der linken Seite wurde die fünfte Kategorie aktualisiert.

HINWEIS: Wenn die in der WHERE-Klausel angegebene Suchbedingung auf mehr als eine Zeile zutrifft, aktualisiert der SQL-Befehl UPDATE alle übereinstimmenden Zeilen.

DATEN AUS EINER TABELLE LÖSCHEN

Der DELETE-SQL-Befehl löscht Datenzeilen aus einer Tabelle. Eine Suchbedingung schränkt ein, welche Zeilen gelöscht werden sollen. Ein JOIN kann verwendet werden, um Daten aus mehreren Tabellen zu löschen.

1. Die folgende SQL-Anweisung verwendet den Befehl DELETE, gefolgt von der FROM-Klausel und dem Namen der Tabelle, aus der die Zeile(n) gelöscht werden soll(en).

2. Als Nächstes gibt die Suchbedingung an, welche Zeilen aus dieser Tabelle gelöscht werden sollen. Im Folgenden wird die zu löschende Zeile durch einen Wert in der Spalte id angegeben.

3. Die ID der zu löschenen Zeile wird in \$id gespeichert. Anschließend wird die ID der Funktion pdo() über ein indiziertes Array übergeben, das im Argument selbst erstellt wird.

```
① $sql = "DELETE FROM category  
        WHERE id = :id;";  
  
②  
③ [ $id = 5;  
    pdo($pdo, $sql, [$id]);
```

Auf der rechten Seite wurde die fünfte Kategorie aus der Tabelle gelöscht.

HINWEIS: Wenn die Suchbedingung in der WHERE-Klausel auf mehr als eine Zeile zutrifft, löscht der SQL-Befehl DELETE alle passenden Zeilen.

category			
id	name	description	navigation
1	Print	Inspiring graphic design	1
2	Digital	Powerful pixels	1
3	Illustration	Hand-drawn visual storytelling	1
4	Photography	Capturing the moment	1

DIE ID EINER NEUEN DATENZEILE ERMITTeln

Wenn eine Spalte einer Datenbanktabelle eine automatisch inkrementierende ID verwendet und der Tabelle eine neue Datenzeile hinzugefügt wird, gibt die Methode `lastInsertId()` des PDO-Objekts die ID zurück, die die Datenbank für die neue Zeile erstellt hat.

In der Beispiel-Website heißt die erste Spalte in jeder Tabelle `id`. Sie enthält einen Primärschlüssel, der zur eindeutigen Identifizierung jeder Zeile der Tabelle dient. Wenn der Tabelle eine neue Zeile hinzugefügt wird, wird mit der Auto-Inkrementierungsfunktion von MySQL der in der `id`-Spalte der neuen Zeile verwendete Wert erzeugt.

Wenn eine SQL-Anweisung ausgeführt wurde, die eine neue Datenzeile in die Tabelle einfügt, kann die Methode `lastInsertId()` des PDO-Objekts den Wert abrufen, den MySQL für die `id`-Spalte erzeugt hat. Dieser kann in einer Variablen gespeichert und später im Code verwendet werden.

Diese Technik wird verwendet, wenn ein neuer Artikel erstellt und gleichzeitig ein Bild hochgeladen wird.

- Das Bild wird zuerst in die Tabelle `image` eingefügt.
- Mit der Methode `lastInsertId()` wird seine ID ermittelt.
- Der Artikel wird der Tabelle `article` zuletzt hinzugefügt, da er die ID des neuen Bilds in der Spalte `image_id` der Tabelle `article` verwendet.

Unten sehen Sie, dass die Methode `lastInsertId()` aufgerufen wird, nachdem die Funktion `pdo()` aufgerufen wurde, um der Datenbank eine neue Zeile hinzuzufügen.

```
pdo($pdo, $sql, $arguments);
$new_id = $pdo->lastInsertId();
```

HERAUSFINDEN, WIE VIELE ZEILEN SICH GEÄNDERT HABEN

Wenn eine SQL-Anweisung einen UPDATE- oder DELETE-Befehl verwendet, kann sie mehr als eine Datenzeile auf einmal ändern. Die Methode rowCount() des PDOStatement-Objekts gibt die Anzahl der betroffenen Zeilen zurück.

Wenn ein UPDATE- oder DELETE-Befehl ausgeführt wird, kann er null, eine oder viele Zeilen in der Datenbank ändern, je nachdem, wie viele Zeilen der Suchbedingung in der WHERE-Klausel entsprechen.

Wenn die folgende Abfrage ausgeführt wird und die Tabelle category keine Kategorie mit der id 100 enthält, wird nichts aus der Datenbank gelöscht:

```
DELETE FROM category  
WHERE id = 100;
```

Wenn die unten stehende Abfrage ausgeführt wird, könnte sie null, eine oder viele Zeilen aktualisieren, je nachdem, wie viele Zeilen den Wert 0 in der Spalte navigation hatten.

```
UPDATE category  
SET navigation = 1  
WHERE navigation = 0;
```

Wenn die Methode execute() des PDOStatement-Objekts aufgerufen wird, gibt sie true zurück, wenn eine SQL-Anweisung ausgeführt wird, und false, wenn dies nicht der Fall ist. Wie die Beispiele zeigen, sagt dies jedoch nichts darüber aus, ob sich Zeilen in der Datenbank geändert haben.

Um festzustellen, wie viele Zeilen geändert wurden, wenn eine SQL-Anweisung ausgeführt wurde, verfügt das PDOStatement-Objekt über die Methode rowCount(), die die Anzahl der geänderten Zeilen zurückgibt.

Die Methode rowCount() sollte in der nächsten Anweisung nach der Methode execute() aufgerufen werden, und der von ihr zurückgegebene Wert kann in einer Variablen gespeichert werden.

Wenn Sie eine SQL-Anweisung mit der im letzten Kapitel definierten pdo()-Funktion ausführen, kann die Methode rowCount() in derselben Anweisung aufgerufen werden, die die pdo()-Funktion aufruft (unter Verwendung der Methodenverkettung).

```
$sql = "UPDATE category  
        SET navigation = 1  
        WHERE navigation = 0;";  
$result = $pdo($pdo, $sql)->rowCount();
```

DOPPELTE WERTE IN SPALTEN VERHINDERN

Die Werte in einigen Spalten sollten eindeutig sein. Auf der Beispiel-Webseite können zwei Artikel nicht denselben Titel haben, zwei Kategorien können nicht denselben Namen haben, und zwei Mitglieder können nicht dieselbe E-Mail-Adresse haben.

Auf S. 430 wurde eine Eindeutigkeitsbedingung zu den folgenden Spalten der Datenbank hinzugefügt, um sicherzustellen, dass keine zwei Zeilen den gleichen Wert in den folgenden Spalten haben:

- Spalte `title` der Tabelle `article`
- Spalte `column` der Tabelle `category`
- Spalte `email` der Tabelle `member`

Wenn eine neue Zeile zu diesen Tabellen hinzugefügt (oder eine bestehende Zeile aktualisiert) wird und eine andere Zeile bereits denselben Wert in diesen Spalten hat, wirft PDO ein Ausnahmeobjekt, da es die Daten nicht speichern kann (die Eindeutigkeitsbedingung würde verletzt).

PDO verwendet die Klasse `PDOException`, um ein `PDOException`-Objekt zu erstellen; es ähnelt den Ausnahmeobjekten, die Sie auf S. 368 kennengelernt haben, enthält aber zusätzliche Daten, die spezifisch für PDO sind. Im Folgenden sehen Sie, wie SQL-Anweisungen behandelt werden, die eine Eindeutigkeitsbedingung aufheben können.

1. Der Code zum Erstellen einer neuen Zeile in diesen Tabellen oder zum Aktualisieren einer bestehenden Zeile wird in einen `try`-Block eingefügt.

2. Wenn PDO bei der Ausführung des Codes im `try`-Block eine Ausnahme auslöst, wird der nachfolgende `catch`-Block ausgeführt und das Ausnahmeobjekt in der Variablen `$e` gespeichert.

3. Das `PDOException`-Objekt hat die Eigenschaft `errorInfo`. Sein Wert ist ein indiziertes Array mit Daten über den Fehler. Das zweite Element in dem Array ist ein Fehlercode (<http://notes.re/PDO/error-codes> enthält eine vollständige Liste der Fehlercodes). Wenn der Fehlercode 1062 lautet, verhindert eine Eindeutigkeitsbedingung das Speichern der Daten, und der Benutzer muss darüber informiert werden, dass der Wert bereits verwendet wurde.

4. Wenn es sich um einen anderen Fehlercode handelt, wird die Ausnahme mit dem Schlüsselwort `throw` (siehe S. 369) erneut ausgelöst und von der Standardfunktion behandelt.

```
① try {  
    pdo($pdo, $sql, $args);  
② } catch (PDOException $e) {  
    ③ if ($e->errorInfo[1] === 1062) {  
        // Wert wurde bereits verwendet  
    } else {  
        throw $e;  
    }  
}
```

WEBSEITEN ZUM BEARBEITEN VON DATENBANKDATEN ERSTELLEN

Nachdem Sie gesehen haben, wie PDO Daten in der Datenbank hinzufügt, aktualisiert und löscht, lernen Sie im restlichen Kapitel, wie Sie Verwaltungsseiten und Formulare erstellen, durch die in einer Datenbank gespeicherte Daten geändert werden können.

Hier sind die sechs Verwaltungsseiten, auf denen die Benutzer Kategorien und Artikel erstellen, aktualisieren und löschen können.

categories.php

Auf dieser Seite sind alle Kategorien aufgelistet. Außerdem gibt es Links zum Erstellen, Aktualisieren und Löschen von Kategorien.

Quelle: CreativeFolk

CATEGORIES	
View more categories	
Digital	Edit Delete
Illustration	Edit Delete
Photography	Edit Delete
Print	Edit Delete

Die Links zum Erstellen oder Bearbeiten der Kategorien verweisen auf category.php.

- Bei der Erstellung einer Kategorie gibt es keine Abfragestring.
- Bei der Bearbeitung einer Kategorie enthält der Abfragestring den Namen id, und sein Wert ist die id der zu bearbeitenden Kategorie. Zum Beispiel: category.php?id=2.

Die Links zum Löschen einer Kategorie verweisen auf die Seite category-delete.php, und der Abfragestring enthält die ID der zu löschen Kategorie.

Die Verwaltungsseiten befinden sich alle im Ordner admin.

Probieren Sie sie im Browser aus, bevor Sie sich den Code ansehen.

articles.php

Auf dieser Seite werden alle Artikel aufgelistet. Sie enthält Links, über die Website-Besitzer Artikel erstellen, aktualisieren und löschen können.

ARTICLES	
View more articles	
MADE	Edit Delete
Title Guide	Apr 26, 2021 Yes
Golden Brown	Apr 25, 2021 Yes

Die Links zum Erstellen oder Bearbeiten der Artikel verweisen auf article.php.

- Bei der Erstellung eines Artikels gibt es keine Abfragestring.
- Bei der Bearbeitung eines Artikels enthält der Abfragestring den Namen id, und sein Wert ist die id des zu bearbeitenden Artikels. Zum Beispiel: article.php?id=2.

Die Links zum Löschen eines Artikels führen zu der Seite article-delete.php, und der Abfragestring enthält die ID des zu löschen Artikels.

category.php

Diese Seite bietet ein Formular, um eine neue Kategorie zu erstellen oder eine bestehende zu aktualisieren.

The screenshot shows a 'Edit Category' form on the 'CREATIVE POLE' website. It includes fields for Name, Description, and Navigation. A validation error message 'Please correct errors' is displayed above the form. Below the fields, a note says 'Description should be 1-254 characters.' A 'SAVE' button is at the bottom right.

Wenn das Formular abgeschickt wird, werden die Daten überprüft:

- Sind sie gültig, aktualisiert die Seite die Datenbank und schickt den Benutzer zurück zur Seite categories.php. Im Abfragestring wird eine Meldung gesendet, damit die Kategorienseite anzeigen kann, dass die Daten gespeichert wurden.
- Sind die Daten ungültig, wird das Formular erneut angezeigt, mit Meldungen unter den Formularfeldern, die korrigiert werden müssen.

category-delete.php

Auf dieser Seite werden die Benutzer aufgefordert, zu bestätigen, dass sie eine Kategorie löschen möchten.

The screenshot shows a 'Delete Category' dialog box. It contains the text 'Click confirm to delete the category: Illustration' and two buttons: 'CONFIRM' (highlighted in red) and 'CANCEL'.

Wenn der Benutzer auf die Schaltfläche confirm klickt, wird die Kategorie gelöscht und der Benutzer zur Seite categories.php zurückgeschickt. Im Abfragestring wird eine Meldung gesendet und auf der Seite categories angezeigt, um den Benutzer darüber zu informieren, dass die Kategorie gelöscht wurde.

article.php

Die Artikelseite bietet ein Formular zum Erstellen eines neuen Artikels oder zum Aktualisieren eines bestehenden Artikels.

The screenshot shows an 'Edit Article' form on the 'CREATIVE POLE' website. It includes fields for Title, Summary, Content, Author, Category, and Published status. A preview image of a cafe interior is shown with the caption 'ALT texts: Photograph of the interior of a cafe'. Buttons for 'EDIT ALT TEXT' and 'DELETE IMAGE' are below the image. A 'PUBLISH' button is at the bottom right.

Wenn das Formular abgeschickt wird, werden die Daten überprüft:

- Sind sie gültig, aktualisiert die Seite die Datenbank und schickt den Benutzer zurück zur Seite articles.php. Im Abfragestring wird eine Nachricht gesendet, sodass die Artikelseite anzeigen kann, dass die Daten gespeichert wurden.
- Sind die Daten ungültig, wird das Formular erneut angezeigt, mit Meldungen unter den Formularfeldern, die korrigiert werden müssen.

article-delete.php

Auf dieser Seite werden die Benutzer aufgefordert zu bestätigen, dass sie einen Artikel löschen wollen.

The screenshot shows a 'Delete Article' dialog box. It contains the text 'Click confirm to delete the article: Stargazing' and two buttons: 'CONFIRM' (highlighted in red) and 'CANCEL'.

Wenn der Benutzer auf die Schaltfläche confirm klickt, wird der Artikel gelöscht und der Benutzer zur Seite articles.php zurückgeleitet. Im Abfragestring wird eine Meldung gesendet und auf der Artikelseite angezeigt, um den Benutzer darüber zu informieren, dass der Artikel gelöscht wurde.

KATEGORIEN ERSTELLEN, AKTUALISIEREN UND LÖSCHEN

Die Seite `categories.php` enthält Links zum Erstellen, Aktualisieren und Löschen von Kategorien.

1. Es wird die strikte Typisierung verwendet, und zwei Dateien werden eingefügt:

`database-connection.php` erstellt ein PDO-Objekt, und `functions.php` enthält benutzerdefinierte Funktionen, darunter `pdo()`, Funktionen zur Datenformatierung und eine neue Funktion, die in den Schritten 15–19 gezeigt wird.

2. Wenn der Abfragestring eine Erfolgsmeldung enthält, wird diese in der Variablen `$success` gespeichert. Wenn nicht, enthält `$success` den Wert `null`.

3. Wenn der Abfragestring eine Fehlermeldung enthält, wird sie in `$failure` gespeichert. Wenn nicht, enthält `$failure` den Wert `null`.

4. Die Variable `$sql` enthält die SQL-Abfrage, um Daten über jede in der Datenbank gespeicherte Kategorie abzurufen.

5. Die Funktion `pdo()` führt die Abfrage aus, und die Methode `fetchAll()` des `PDOStatement`-Objekts nimmt die Kategoriedaten und speichert sie in `$categories`.

6. Die Header-Datei für die Admin-Seiten wird eingefügt.

7. Wenn der Abfragestring eine Erfolgs- oder Fehlermeldung enthielt, wird diese auf der Seite angezeigt.

8. Ein Link zum Erstellen einer neuen Kategorie wird hinzugefügt. Wenn ein Link zu `category.php` keinen Abfragestring enthält, weiß `category.php`, dass es eine neue Kategorie erstellen soll.

9. Eine Tabelle wird der Seite hinzugefügt. Die erste Zeile enthält drei Spaltenüberschriften: `name`, `edit` und `delete`.

10. Mit einer `foreach`-Schleife werden Daten über bestehende Kategorien mit Links zum Bearbeiten oder Löschen angezeigt.

11. Die erste Spalte zeigt den Namen der Kategorie an.

12. Anschließend wird ein Link zur Seite `category.php` erstellt.

Der Abfragestring enthält die ID der Kategorie, sodass der Benutzer beim Laden der Seite `category.php` die Details zu dieser Kategorie bearbeiten kann. Zum Beispiel: ``.

13. Es wird ein Link zu der Seite `category-delete.php` erstellt, die eine Kategorie aus der Datenbank löscht. Die ID der Kategorie wird im Abfragestring gespeichert.

14. Die Fußzeile für die Verwaltungsseiten wird eingefügt.

15. Eine neue Funktion, die Benutzer auf eine andere Seite umleitet, wurde in die `functions.php` eingefügt. Sie ermöglicht das Hinzufügen von Erfolgs- oder Fehlermeldungen in den Abfragestring der Seite, zu der der Benutzer weitergeleitet wird. Sie hat drei Parameter:

- Name der Datei, zu der weitergeleitet wird
- optionales Array, das zur Erstellung eines Abfragestrings verwendet wird
- optionaler HTTP-Antwortcode (Standard ist 302)

16. Die Variable `$qs` wird erstellt, um einen Abfragestring zu speichern.

Ihr Wert wird mit einem ternären Operator zugewiesen. Wenn `$parameters` ein Array enthält, wird ein Fragezeichen an `$qs` angehängt, und die in PHP integrierte Funktion `http_build_query()` erstellt den Abfragestring aus den Werten im Array. Für jedes Element im Array wird der Schlüssel zu einem Namen im Abfragestring und sein Wert nach einem Gleichheitszeichen hinzugefügt (Zeichen, die in einer URL nicht erlaubt sind, werden ebenfalls escaped – siehe Seite 280).

17. Der Wert in `$qs` wird an das Ende der URL der Seite angehängt, zu der der Benutzer geschickt wird.

18. Die Funktion `header()` von PHP wird aufgerufen, um den Besucher umzuleiten. Das erste Argument teilt dem Browser die anzufordernde Seite mit; das zweite ist der HTTP-Antwortcode.

19. `exit` verhindert, dass weiterer Code ausgeführt wird.

```

<?php
declare(strict_types = 1);
① include '../includes/database-connection.php';
include '../includes/functions.php';

② $success = $_GET['success'] ?? null; // Prüfung Erfolgsmeldung
③ $failure = $_GET['failure'] ?? null; // Prüfung Fehlermeldung

④ $sql = "SELECT id, name, navigation FROM category;"; // SQL alle Kategorien
⑤ $categories = pdo($pdo, $sql)->fetchAll(); // Alle Kategorien
?>

⑥ <?php include '../includes/admin-header.php' ?>
<main class="container" id="content">
    <section class="header">
        <h1>Categories</h1>
    ⑦ <?php if ($success) { ?><div class="alert alert-success"><?= $success ?></div><?php } ?>
    <?php if ($failure) { ?><div class="alert alert-danger"><?= $failure ?></div><?php } ?>
    ⑧ <p><a href="category.php" class="btn btn-primary">Add new category</a></p>
</section>

    <table class="categories">
        ⑨ <tr><th>Name</th><th class="edit">Edit</th><th class="delete">Delete</th></tr>
        ⑩ <?php foreach ($categories as $category) { ?>
            <tr>
                ⑪ <td><?= html_escape($category['name']) ?></td>
                ⑫ <td><a href="category.php?id=<?= $category['id'] ?>">
                    class="btn btn-primary">Edit</a></td>
                ⑬ <td><a href="category-delete.php?id=<?= $category['id'] ?>">
                    class="btn btn-danger">Delete</a></td>
            </tr>
        ⑭ <?php } ?>
    </table>
</main>
⑯ <?php include '../includes/admin-footer.php'; ?>

```

```

⑮ function redirect(string $location, array $parameters = [], $response_code = 302)
{
    ⑯ $qs = $parameters ? '?' . http_build_query($parameters) : ''; // Querystring
    $location = $location . $qs; // Neuer Pfad
    ⑰ header('Location: ' . $location, $response_code); // Redirect neue Seite
    ⑱ exit; // Anhalten
}

```

DATEN ERSTELLEN UND AKTUALISIEREN

Der Code zum Erstellen oder Aktualisieren von Artikeln und Kategorien ist in vier Teile unterteilt, jeder mit mehreren if-Anweisungen, die bestimmen, welcher Code ausgeführt wird.

A: SEITE EINRICHTEN

Zunächst prüfen die Seiten, ob Daten erstellt oder aktualisiert werden. Dazu prüfen sie, ob der Abfragestring den Namen `id` mit einem ganzzahligen Wert hat.

- Nein: Mit der Seite wird eine neue Zeile in der Datenbank erstellt. An dieser Stelle geht der PHP-Interpreter zu Teil B über.
- Ja: Die Seite versucht, eine bestehende Datenzeile zu bearbeiten, und muss diese Daten laden, damit sie bearbeitet werden können.

Wenn die zu bearbeitenden Daten nicht zurückgegeben werden, wird dem Benutzer mitgeteilt, dass der Artikel oder die Kategorie nicht gefunden wurde.

B: BENUTZERDATEN ABRUFEN UND VALIDIEREN

Als Nächstes prüft die Seite, ob das Formular abgeschickt wurde.

- Nein: Es geht mit Teil D weiter.
- Ja: Die Daten müssen abgerufen und validiert werden.

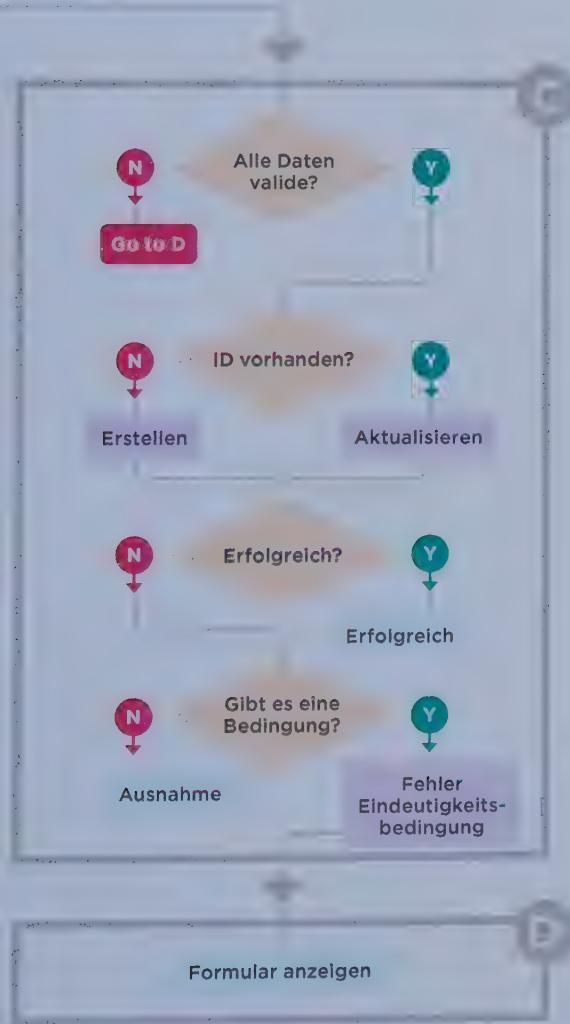
Es wird ein Array mit einem Element für jedes Daten-element auf der Seite erstellt. Der Wert für dieses Element wird zugewiesen, indem die Daten mithilfe der in Kapitel 6 vorgestellten Funktionen überprüft werden.

- Wenn die Daten gültig sind: Das Element im Array enthält eine leere Zeichenkette.
- Wenn die Daten ungültig sind: Das Array speichert eine Fehlermeldung, die angibt, welche Daten von dem Steuerelement erwartet werden.

Dann werden die Werte im Array zu einer Zeichenkette zusammengefasst.



Mit Flussdiagrammen kann beschrieben werden, welcher Code in den verschiedenen Situationen ausgeführt werden soll, und Sie können bei der Arbeit am Code auf diese Flussdiagramme zurückgreifen.



C: BENUTZERDATEN SPEICHERN

In diesem Teil prüft die Seite, ob alle Daten gültig sind.

- Nein: Weiter mit Teil D.
- Ja: Fahre mit dem Code in Teil C fort.

Dann wird geprüft, ob der Abfragestring eine ID enthielt:

- Nein: Der SQL-Code erstellt einen neuen Artikel oder eine neue Kategorie.
- Ja: Der SQL-Code aktualisiert den Artikel oder die Kategorie.

Als Nächstes wird geprüft, ob der SQL-Code erfolgreich ausgeführt wurde:

- Nein: Überprüfe die Art der ausgelösten Ausnahme.
- Ja: Eine Erfolgsmeldung wird angezeigt.

Wenn eine Ausnahme ausgelöst wurde, prüfen Sie, ob sie durch eine Eindeutigkeitsbedingung verursacht wurde:

- Nein: Die Ausnahme wird erneut ausgelöst.
- Ja: Es wird eine Meldung angezeigt, in der das Problem beschrieben wird.

D: FORMULAR ANZEIGEN

Nun wird das Formular angezeigt:

- Wenn es keine ID gibt und das Formular nicht abgeschickt wurde, ist das Formular leer.
- Wenn eine ID vorhanden ist, das Formular aber nicht abgeschickt wurde, zeigt das Formular die vorhandenen Daten an, die bearbeitet werden müssen.
- Wenn das Formular abgeschickt wurde und die Daten ungültig sind, zeigt das Formular eingegebenen Daten an, mit Fehlermeldungen, die erklären, wie sie korrigiert werden können.

KATEGORIEDATEN ABRUFEN UND VALIDIEREN

Der Code für category.php ist auf die folgenden sechs Seiten aufgeteilt. Zunächst sehen Sie den Code für die Teile A und B (wie auf der vorherigen Seite beschrieben). Teil A richtet die Seite ein und bestimmt, ob eine neue Kategorie erstellt oder eine bestehende aktualisiert wird.

1. Die strikte Typisierung ist aktiviert, und die erforderlichen Dateien werden eingefügt. Es werden die Dateien database-connection.php, functions.php und validate.php (mit der in Kapitel 6 erstellten Überprüfungsfunktionen) benötigt.

2. Wenn die Seite eine bestehende Kategorie bearbeitet, enthält die URL einen Abfragestring mit dem Namen id; der Wert ist die ID der zu bearbeitenden Kategorie.

Mit der PHP-Funktion filter_input() wird geprüft, ob eine id vorhanden und ihr Wert eine ganze Zahl ist. Die Variable \$id speichert:

- die ganze Zahl, wenn vorhanden
- false, wenn der Wert keine gültige ganze Zahl ist
- null, wenn der Name id nicht im Abfragestring enthalten ist

3. Das Array \$category wird deklariert, um Details über die Kategorie zu speichern. Es wird mit Werten initialisiert, die im Formular in Teil D (siehe S. 503) angezeigt werden können, wenn eine neue Kategorie erstellt wird und das Formular noch keine Werte zur Anzeige enthält.

4. Das Array \$errors wird mit leeren Strings für jedes Element initialisiert (weil noch keine Fehler entdeckt wurden). Die Werte in diesem Array werden in Teil D nach jedem Formularsteuerelement angezeigt (siehe S. 503).

5. Eine if-Anweisung prüft, ob eine gültige Ganzzahl für die Kategorie-ID im Abfragestring vorhanden ist.

6. Wenn ja, wird die SQL-Anweisung für die Kategorie, die der Benutzer bearbeiten möchte, in \$sql gespeichert.

7. Die Funktion pdo() führt die Abfrage aus, und die Methode fetch() sammelt die Kategoriedaten.

Das zurückgegebene Array wird in der Variablen \$category gespeichert (wobei die in Schritt 3 erstellten Werte überschrieben werden).

8. Wenn der Abfragestring eine ID enthieilt, die Datenbank aber keine passende Kategorie gefunden hat, wird die Variable \$category auf false gesetzt, und der nachfolgende Codeblock wird ausgeführt.

9. Die Funktion redirect() (S. 495) leitet den Benutzer zur Seite categories.php weiter. Das zweite Argument ist ein Array, das verwendet wird, um eine Fehlermeldung anzuzeigen, die dem Benutzer mitteilt, dass die Kategorie nicht gefunden werden konnte.

Teil B sammelt und validiert alle Formulardaten:

10. Eine if-Anweisung testet, ob das Formular abgeschickt wurde.

11. Wenn ja, werden die Formularwerte im Array \$category gespeichert, das in Schritt 4 erstellt wurde. Hinweis: Die Navigationsoption (die angibt, ob die Kategorie in der Navigationsleiste angezeigt werden soll oder nicht) wird nur an den Server gesendet, wenn das Kontrollkästchen aktiviert ist. Daher wird mit der PHP-Funktion isset() geprüft, ob ein Wert für dieses Formularsteuerelement gesendet wurde, und dann wird mit einem Gleichheitsoperator geprüft, ob der Wert 1 ist. Wenn ja, enthält die Navigationstaste den Wert 1; wenn nicht, enthält sie 0.

12. Der Kategorienname und die Beschreibung werden mit der Funktion is_text() in der Include-Datei validiert. Sind sie ungültig, werden Fehlermeldungen im Array \$errors gespeichert.

13. Die Werte im Array \$errors werden zusammengefügt und in der Variablen \$invalid gespeichert.

Auf der nächsten Seite sehen Sie, dass die Seite entscheidet, ob die Daten in der Datenbank gespeichert werden sollen oder nicht.

```
<?php
// Part A: Setup
declare(strict_types = 1); // Strikte Typisierung
① include '../includes/database-connection.php'; // Datenbankverbindung
include '../includes/functions.php'; // Funktionen einfügen
include '../includes/validate.php';

// Initialize variables
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // ID abrufen & valid.
$category = [
    'id'          => $id,
    'name'        => '',
    'description' => '',
    'navigation'  => false,
];
// Kategorie-Array initialisieren
③ $errors = [
    'warning'    => '',
    'name'        => '',
    'description' => '',
];
// Fehler-Array initialisieren
④

// Wenn ID vorhanden, wird aktuelle Kategorie ermittelt
⑤ if ($id) { // wenn ID vorhanden
    $sql = "SELECT id, name, description, navigation
            FROM category
            WHERE id = :id;"; // SQL-Anweisung
    ⑥ $category = pdo($pdo, $sql, [$id])->fetch(); // Kategorie-Daten abfragen
    ⑦ if (!$category) { // Wenn keine Kategorie gefunden wurde
        redirect('categories.php', ['failure' => 'Category not found']); // Fehler
    }
}
// Teil B: Abrufen und Überprüfen der Formulardaten
⑧ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular abgeschickt
    $category['name']      = $_POST['name']; // Name abrufen
    $category['description'] = $_POST['description']; // Beschreibung
    $category['navigation'] = (isset($_POST['navigation'])
        and ($_POST['navigation'] == 1)) ? 1 : 0; // Navigation
}
// Prüfen, ob alle Daten gültig sind; anderenfalls Fehlermeldungen
⑨ $errors['name'] = (is_text($category['name'], 1, 24))
    ? '' : 'Name should be 1-24 characters.'; // Name
$errors['description'] = (is_text($category['description'], 1, 254))
    ? '' : 'Description should be 1-254 characters.'; // Beschreibung
⑩
⑪
⑫
⑬ $invalid = implode($errors); // Fehlermeldungen
```

KATEGORIE-DATEN SPEICHERN

In Teil C dieser Seite wird in der Datei category.php festgelegt, ob die Datenbank die Daten speichern soll oder nicht und, wenn ja, ob eine neue Kategorie hinzugefügt oder eine bestehende Kategorie aktualisiert werden soll.

1. Eine if-Anweisung prüft, ob \$invalid Text enthält. Ist dies der Fall, wird die Bedingung als true ausgewertet, was bedeutet, dass es Fehler gibt, die der Benutzer korrigieren muss. Der folgende Codeblock speichert eine Warnmeldung im Array \$errors.

2. Andernfalls sind die Daten gültig und können verarbeitet werden.

3. Die Daten aus dem Array \$category werden in die Variable \$arguments kopiert. Dies geschieht aus folgenden Gründen:

Wenn die Funktion pdo() die SQL-Anweisung ausführt, verwendet sie die Werte, die in \$category gespeichert wurden, um die Platzhalter zu ersetzen.

Die SQL-Anweisung benötigt jedoch nicht immer alle Elemente, die im Array \$category gespeichert sind, und die Funktion pdo() wird nicht ausgeführt, wenn einige der Elemente nicht entfernt werden (siehe Schritt 9).

4. Wenn \$id eine Zahl enthält (was als true behandelt wird), bedeutet dies, dass die Seite eine bestehende Kategorie aktualisiert.

5. Die Variable \$sql enthält die SQL-Anweisung zur Aktualisierung der Kategorie. Sie beginnt mit dem Befehl UPDATE und dem Namen der zu aktualisierenden Tabelle.

6. Auf die SET-Klausel folgen die Namen der zu aktualisierenden Spalten und Platzhalter, die durch die Werte dieser Spalten ersetzt werden.

7. Die WHERE-Klausel gibt die ID der Zeile in der Tabelle category an, die aktualisiert werden soll.

8. Wenn \$id keine Zahl enthält, bedeutet dies, dass die Seite eine neue Kategorie zur Datenbank hinzufügt.

9. Die Funktion unset() von PHP entfernt das Element mit der Artikel-ID aus dem Array \$arguments. Dies geschieht, weil das Daten-Array, das die Werte enthält, die zum Ersetzen der Platzhalter in der SQL-Anweisung verwendet werden, keine zusätzlichen Elemente enthalten darf.

10. Die Variable \$sql enthält die SQL-Anweisung zur Erstellung einer neuen Kategorie. Sie beginnt mit:

- INSERT, um eine neue Zeile in die Datenbank einzufügen
- INTO, gefolgt von dem Namen der Tabelle, der die Daten hinzugefügt werden sollen
- den Namen der Spalten, die mit Werten versehen werden sollen, in Klammern

11. Auf den Befehl VALUES folgen die Namen der Platzhalter, die die neuen Werte repräsentieren. Sie werden ebenfalls in Klammern geschrieben.

12. Die SQL-Anweisung wird in einem try-Block ausgeführt, da der Name der Kategorie einer Eindeutigkeitsbedingung unterliegt und eine Ausnahme ausgelöst würde, wenn der Benutzer versuchen würde, einen Namen einzugeben, der bereits verwendet wurde.

13. Die Funktion pdo() führt die SQL-Anweisung aus.

14. Wenn der Code im try-Block noch läuft, wurde die SQL-Anweisung erfolgreich ausgeführt, sodass die redirect()-Funktion (siehe S. 495) aufgerufen wird. Das erste Argument gibt an, dass der Benutzer zur Seite categories.php zurückgeschickt werden soll. Das zweite Argument ist ein Array, das anzeigt, dass die Kategorie gespeichert wurde:

```
['success'=> 'Category saved']
```

Die Funktion redirect() nimmt diese Daten und weist den Browser an, die folgende Seite aufzurufen: category.php?success=Category%20saved

```

// Teil C: Prüfen, ob Daten valide, wenn ja -> Datenbank aktualisieren
if ($invalid) {                                     // Wenn Daten ungültig
    $errors['warning'] = 'Please correct errors';   // Fehlermeldung
} else {
    $arguments = $category;                         // Argument-Array für SQL
    if ($id) {                                      // Wenn id vorhanden
        $sql = "UPDATE category
                  SET name = :name, description = :description,
                      navigation = :navigation
                 WHERE id = :id;";                     // SQL-Kategorie aktualisieren
    } else {                                         // Wenn keine id
        unset($arguments['id']); // id aus category-Array löschen
        $sql = "INSERT INTO category (name, description, navigation)
                  VALUES (:name, :description, :navigation);"; // Kategorie erst.
    }

    // Bei der SQL-Ausführung können drei Dinge geschehen:
    // Kategorie gespeichert | Name schon verwendet | Ausnahme aus anderem Grund
    try {
        pdo($pdo, $sql, $arguments);                // try-Block Beginn
        redirect('categories.php', ['success' => 'Category saved']); // Redirect
    } catch (PDOException $e) {                     // Wenn PDO-Ausnahme
        if ($e->errorInfo[1] == 1062) {             // Wenn Dublette
            $errors['warning'] = 'Category name already in use'; // Fehlermeldung
        } else {                                    // Sonst unerwarteter Fehler
            throw $e;                            // Ausnahme erneut
        }
    }
}
?>

```

Durch die Weiterleitung des Besuchers zu categories.php, wenn die Kategorie gespeichert wurde, müssen die Daten nicht erneut eingegeben werden beziehungsweise die Seite muss nicht aktualisiert werden.

15. Wenn die Kategoriedaten nicht gespeichert werden konnten, wird eine Ausnahme ausgelöst, und der PHP-Interpreter führt den Code im catch-Block aus.

Der Zweck des catch-Blocks ist es, zu prüfen, ob die Ausnahme ausgelöst wurde, weil der Kategorienname nicht eindeutig war. Innerhalb des catch-Blocks wird das Ausnahmeobjekt in der Variablen \$e gespeichert.

16. Die Bedingung einer if-Anweisung prüft die Eigenschaft `errorInfo` des Ausnahmeobjekts, das ein indiziertes Array enthält. Ein Fehlercode wird in dem Element gespeichert, dessen Schlüssel 1 ist. Der Fehlercode 1062 bedeutet, dass die Eindeutigkeitsbedingung verletzt wurde und der Kategorienname bereits verwendet wird.

17. Eine Fehlermeldung wird im Array `$errors` gespeichert, die dem Benutzer mitteilt, dass der Kategorienname bereits in Gebrauch ist.

18. Wenn das Exception-Objekt einen anderen Fehlercode hat, wird die Exception erneut ausgelöst und von der Standard-Exception-Handling-Funktion behandelt.

FORMULAR ZUM ERSTELLEN/BEARBEITEN VON KATEGORIEDATEN

Teil D des Prozesses besteht darin, dem Besucher ein Formular zu zeigen, das er zum Erstellen oder Bearbeiten von Kategorieinformationen verwenden kann. Egal ob der Benutzer eine Kategorie erstellt oder bearbeitet, ihm wird immer das gleiche Formular angezeigt.

1. Das Attribut `action` des öffnenden `<form>`-Tags verweist auf die gleiche Seite (`category.php`).

Der Abfragestring enthält den Schlüssel `id`, und sein Wert ist der Wert, der im Attribut `$id` gespeichert ist. Wenn die Kategorie bereits erstellt wurde, enthält er die ID der Kategorie; andernfalls ist er `null`. Das Formular wird mit HTTP POST gesendet.

2. Wenn das Formular abgeschickt wurde und die Daten nicht gültig waren oder wenn der Kategorienname bereits verwendet wurde, wird eine Fehlermeldung im Array `$errors` als Wert für den Schlüssel `warning` gespeichert. Eine `if`-Anweisung prüft, ob eine Fehlermeldung vorhanden ist.

3. Wenn ja, wird sie über dem Formular angezeigt.

4. Eine Texteingabe ermöglicht es dem Benutzer, den Kategorienamen einzugeben oder zu aktualisieren. Wenn bereits ein Name eingegeben wurde, wird dieser im `value`-Attribut der Texteingabe angezeigt.

Die Funktion `html_escape()` sorgt dafür, dass alle reservierten HTML-Zeichen in diesem Wert durch Entities ersetzt werden. Dadurch wird das Risiko eines XSS-Angriffs vermieden.

Wenn die Seite zum ersten Mal geladen wird, um eine neue Kategorie zu erstellen, hat der Benutzer noch keine Kategoriedaten eingegeben.

Deshalb ist es wichtig, das Array `$category` in Teil A zu initialisieren (indem Sie die Schlüsselnamen angeben und ihre Werte auf leere Strings setzen), damit die Seite über Werte verfügt, die sie in den Formularsteuerelementen anzeigen kann.

5. Das Array `$errors` wurde ebenfalls in Teil A initialisiert. Es enthält ein Element für jede Texteingabe. Wenn das Formular abgeschickt wurde und der Kategorienname nicht gültig war, enthält der Wert, der mit dem Schlüssel `name` verbunden ist, eine Fehlermeldung, die das Problem beschreibt, und diese wird unter der Texteingabe angezeigt.

Wenn das Formular nicht abgeschickt wurde oder keine Fehler aufraten, enthält es eine leere Zeichenkette (weil es in Teil A initialisiert wurde), und die leere Zeichenkette wird unter der Texteingabe angezeigt. (Wäre das Array `$errors` in Teil A nicht initialisiert worden, führte der Versuch, diese Meldung anzulegen, zu einem `Undefined index error`.)

6. Eine `<Text area>`-Eingabe ermöglicht es dem Benutzer, eine Beschreibung für die Kategorie einzugeben. Wurde bereits ein Wert eingegeben, wird er zwischen dem öffnenden und dem schließenden Tag angezeigt.

7. Wenn es ein Problem bei der Validierung der Beschreibung gab, wird die Fehlermeldung darunter angezeigt.

8. Ein Kontrollkästchen gibt an, ob der Name der Kategorie in der Navigation angezeigt werden soll oder nicht.

9. Ein ternärer Operator prüft, ob der Navigationsoption der Wert 1 zugewiesen wurde. Wenn ja, fügt er das Attribut `checked` zur Eingabe des Kontrollkästchens hinzu, um es auszuwählen. Wenn nicht, wird stattdessen eine leere Zeichenkette ausgegeben.

10. Am Ende des Formulars wird eine `submit`-Schaltfläche eingefügt.

```
<?php include 'includes/admin-header.php'; ?>
<main class="container admin" id="content">
①   <form action="category.php?id=<?= $id ?>" method="post" class="narrow">

    <h2>Edit Category</h2>
②    <?php if ($errors['warning']) { ?>
        <div class="alert alert-danger"><?= $errors['warning'] ?></div>
    <?php } ?>

    <div class="form-group">
        <label for="name">Name: </label>
        <input type="text" name="name" id="name"
            value="<?= html_escape($category['name']) ?>" class="form-control">
        <span class="errors"><?= $errors['name'] ?></span>
    </div>

    <div class="form-group">
        <label for="description">Description: </label>
        <textarea name="description" id="description" class="form-control">
            <?= html_escape($category['description']) ?></textarea>
        <span class="errors"><?= $errors['description'] ?></span>
    </div>

    <div class="form-check">
        <input type="checkbox" name="navigation" id="navigation"
            value="1" class="form-check-input"
            <?= ($category['navigation'] == 1) ? 'checked' : '' ?>>
        <label class="form-check-label" for="navigation">Navigation</label>
    </div>

⑩    <input type="submit" value="save" class="btn btn-primary btn-save">

    </form>
</main>
<?php include 'includes/admin-footer.php'; ?>
```

EINE KATEGORIE LÖSCHEN

Wenn der Benutzer auf den Link zum Löschen einer Kategorie klickt, ruft die Seite den Kategorienamen aus der Datenbank ab und zeigt ihn dem Benutzer an, wobei er aufgefordert wird zu bestätigen, dass er die Kategorie löschen möchte (dies verhindert, dass jemand versehentlich auf einen Link klickt, der eine Kategorie löscht).

Bestätigt der Benutzer, dass er die Kategorie löschen möchte, wird die Seite neu geladen und versucht, sie zu löschen.

Gelingt dies, wird der Benutzer zu categories .php weitergeleitet und erhält eine Meldung, dass es funktioniert hat.

1. Die strikte Typisierung ist aktiviert und die erforderlichen Dateien werden eingefügt.
2. filter_input() prüft, ob der Name id im Abfragestring enthalten ist. Wenn er vorhanden ist und eine gültige Ganzzahl enthält, wird er in \$id gespeichert. Wenn der Wert keine gültige Ganzzahl ist, wird \$id als false gespeichert. Ist er nicht vorhanden, speichert \$id null.

3. Die Variable \$category enthält den Namen der Kategorie; sie wird mit einem leeren String initialisiert.
4. Eine if-Anweisung prüft, ob der Wert in \$id nicht true ist (wenn er in Schritt 2 auf false oder null gesetzt wurde). Ist dies der Fall, wird zu categories .php weitergeleitet und eine Meldung ausgegeben, dass die Kategorie nicht gefunden wurde.
5. Wenn die Seite noch läuft, speichert die Variable \$sql eine SQL-Abfrage, um den Kategorienamen zu ermitteln.
6. Die Funktion pdo() führt die SQL-Abfrage aus, und die Methode fetchColumn() versucht, den Namen der Kategorie zu ermitteln. Wenn die Kategorie gefunden wurde, wird ihr Name in \$category gespeichert. Wenn nicht, gibt fetchColumn() false zurück.

- 7.** Eine if-Anweisung prüft, ob der Wert in \$category falsch ist. Ist dies der Fall, wird zu categories .php weitergeleitet und eine Meldung ausgegeben, dass die Kategorie nicht gefunden wurde.
- 8.** Wenn das Formular abgeschickt wurde ...
- 9.** Ein try-Block wird erstellt, um die Kategorie zu löschen.
- 10.** Der SQL-Code zum Löschen der Kategorie wird in \$sql gespeichert.
- 11.** Mit der Funktion pdo() wird nun die Kategorie gelöscht.
- 12.** Wenn die SQL-Anweisung ohne Fehler abläuft, wird mit der redirect() -Funktion der Benutzer zu categories .php geschickt, mit einer Nachricht, die bestätigt, dass die Kategorie gelöscht wurde.
- 13.** Wenn bei der Ausführung der SQL-Anweisung eine Ausnahme ausgelöst wurde, wird der catch-Block ausgeführt.
- 14.** Wenn der Fehlercode 1451 ist, verhindert eine Integritätsbedingung das Löschen der Kategorie (weil die Kategorie noch Artikel enthält).
- 15.** In diesem Fall wird mit der Funktion redirect() auf die Seite categories .php weitergeleitet, mit der Fehlermeldung, dass die Kategorie Artikel enthält, die erst verschoben oder gelöscht werden müssen.
- 16.** Andernfalls wird der Fehler erneut ausgelöst und durch den Standard-Exception-Handler behandelt.
- 17.** Über das Formular wird der Kategorienname angezeigt und der Benutzer aufgefordert zu bestätigen, dass die Kategorie gelöscht werden soll. Das Attribut action des <form>-Tags sendet das Formular an category-delete .php. Der Abfragestring enthält die ID der zu löschen Kategorie.
- 18.** Der Name der zu löschen Kategorie wird angezeigt.
- 19.** Mit der submit-Schaltfläche wird bestätigt, dass die Kategorie gelöscht werden kann.

```
<?php
declare(strict_types = 1); // Strikte Typisierung
① include '../includes/database-connection.php'; // Datenbankverbindung
include '../includes/functions.php'; // Funktionen einfügen

② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
③ $category = ''; // Kategoriename initialisieren

④ if (!$id) { // Wenn ID nicht valide
    redirect('categories.php', ['failure' => 'Category not found']); // Umleitung + Fehler
}

⑤ $sql = "SELECT name FROM category WHERE id = :id;"; // Abfrage Kategoriename
⑥ $category = pdo($pdo, $sql, [$id])->fetchColumn();
⑦ if (!$category) { // Wenn keine Kategorie
    redirect('categories.php', ['failure' => 'Category not found']); // Umleitung + Fehler
}

⑧ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular abgeschickt
    try {
        $sql = "DELETE FROM category WHERE id = :id;"; // Kategorie löschen
        pdo($pdo, $sql, [$id]);
        redirect('categories.php', ['success' => 'Category deleted']); // Umleitung
    } catch (PDOException $e) { // Ausnahme abfangen
        if ($e->errorInfo[1] == 1451) { // Wenn Integritätsbedingung
            redirect('categories.php', ['failure' => 'Category contains articles that
                must be moved or deleted before you can delete it']); // Umleitung
        } else { // Sonst
            throw $e; // Ausnahme erneut auslösen
        }
    }
}?
<?php include 'includes/admin-header.php'; ?>

<main class="container admin" id="content">
    <h2>Delete Category</h2>
    <form action="category-delete.php?id=<?= $id ?>" method="POST" class="narrow">
        <p>Click confirm to delete the category <?= html_escape($category)
    ?></p>
        <input type="submit" name="delete" value="confirm" class="btn btn-primary">
        <a href="categories.php" class="btn btn-danger">cancel</a>
    </form>
</main>

<?php include 'includes/admin-footer.php'; ?>
```

ARTIKEL ERSTELLEN UND BEARBEITEN

Die Datei `article.php`, die zum Erstellen und Bearbeiten von Artikeln verwendet wird, hat einen sehr ähnlichen Kontrollfluss wie die Datei `category.php`, muss aber mehr Daten erfassen und erlaubt es auch, Bilder hochzuladen, was die Komplexität erhöht.

Die Datei `article.php` ermöglicht den Benutzern:

- Text für einen Artikel zu erstellen oder bearbeiten
 - ein Artikelbild hochzuladen
- Sie ist komplexer als `category.php`, weil
- zu jedem Artikel mehr Daten gespeichert werden, sodass mehr Daten abgerufen und überprüft werden müssen
 - die Artikeldaten mit Daten in anderen Tabellen verknüpft sind (Kategorie und Mitglied, das den Artikel verfasst hat)
 - die Nutzer optional ein Bild für den Artikel mit einem beschreibenden Alt-Text hochladen können

Um einen Artikel und sein Bild in der Datenbank zu speichern, muss der PHP-Code sowohl mit der `article`-Tabelle als auch mit `image`-Tabelle arbeiten.

Da SQL immer nur neue Datenzeilen in eine Tabelle einfügen kann, verwenden die SQL-Anweisungen zum Erstellen und Bearbeiten eines Artikels **Transaktionen**. Mithilfe von Transaktionen kann die Datenbank prüfen, ob die Änderungen in einer Reihe von SQL-Anweisungen erfolgreich ausgeführt werden können, und sie speichert die Änderungen nur, wenn dies bei allen der Fall ist. Gibt es ein Problem mit nur einer SQL-Anweisung in der Transaktion, wird keine der Änderungen gespeichert.

Wie Sie gesehen haben, bestimmt der Kontrollfluss, welche Anweisungen ausgeführt werden. Die Möglichkeit, Bilder hochzuladen, könnte den Kontrollfluss noch viel komplexer machen. Stellen Sie sich vor, ein Benutzer hat bereits einen Artikel und ein Bild hochgeladen und möchte nun diese Daten bearbeiten:

- Den Artikel aktualisieren, aber das Bild unverändert lassen: Nur die Tabelle `article` würde aktualisiert.
- Den Artikel und das Bild aktualisieren. Hier müssten zuerst die alte Bilddatei und die entsprechenden Daten aus den Tabellen `image` und `article` gelöscht werden, dann die neue Bilddatei hochgeladen und anschließend sowohl die Tabelle `article` als auch die Tabelle `image` mit den neuen Daten aktualisiert werden.
- Nur den alt-Text des Bilds ändern. Das würde bedeuten, dass nur die Tabelle `image` aktualisiert wird.

Je mehr Optionen der Benutzer auf einer einzigen Seite hat, desto komplexer wird der Kontrollfluss und umso schwieriger wird es, dem Code zu folgen.

Um zu verhindern, dass der Kontrollfluss zu komplex wird, können Sie die Anzahl der Aktionen einschränken, die ein Benutzer auf einer einzelnen Seite durchführen kann. So müssen Benutzer beispielsweise ein Bild löschen, bevor sie ein neues hochladen können, und es gibt eine separate Seite zur Bearbeitung des Alt-Texts.

Beim Erstellen eines Artikels können die Benutzer gleichzeitig ein Bild und einen Text eingeben.

HINWEIS: Die meisten Browser erlauben keinen serverseitigen Code, um das HTML-Steuerelement zum Hochladen von Dateien zu füllen. Wenn ein Artikel eingereicht wird, er aber die Validierung nicht besteht, müssen die Benutzer daher das Bild erneut auswählen.

Wenn die Benutzer das Bild erneut hochladen müssen, sind sie auch gezwungen, den Alt-Text erneut einzugeben.

Aktualisieren eines Artikels:

- Wenn der Benutzer kein Bild hochgeladen hat, wird der Teil des Formulars angezeigt, in dem er ein Bild hochladen kann (wie oben).
- Wenn er ein Bild hochgeladen hat, werden ihm das Bild und dessen Alt-Text angezeigt und nicht das Formular.
- Sobald ein Bild hochgeladen wurde, befinden sich unter dem Bild zwei Links, die es den Nutzern ermöglichen, den Alt-Text zu bearbeiten und das Bild (und den Alt-Text) zu löschen.

Die Seite `articles.php`, die alle Artikel auflistet, ist im Code-Download enthalten und funktioniert genauso wie die Seite `categories.php`, die Sie auf S. 494–495 kennengelernt haben.

CREATIVE POLE

[Articles / Categories](#)

Edit Article

Upload image:

No file chosen

Alt text:

Title:

Summary:

Content:

Author:

Category:

Published

CREATIVE POLE

[Articles / Categories](#)

Edit Article

Image:

Alt text:

[EDIT ALT TEXT](#) [DELETE IMAGE](#)

Title:

Summary:

Content:

Author:

Category:

Published

Auf den nächsten beiden Seiten werden Transaktionen vorgestellt, die in `article.php` verwendet werden.

Es dauert dann acht Seiten, um zu zeigen, wie `article.php` funktioniert, da über 230 Zeilen Code enthalten sind.

TRANSAKTIONEN: MEHRERE SQL- ANWEISUNGEN

Mit einer Transaktion wird eine Reihe von SQL-Anweisungen zusammengefasst. Wenn alle Anweisungen erfolgreich sind, werden alle Änderungen in der Datenbank gespeichert. Schlägt eine der Anweisungen fehl, wird keine der Änderungen gespeichert.

Einige Aufgaben erfordern mehr als eine SQL-Anweisung. Wird zum Beispiel ein Bild für einen Artikel hochgeladen, muss der PHP-Code:

- Bilddaten zur Tabelle `image` hinzufügen
- die ID abrufen, die die Datenbank für das Bild erstellt hat (diese wird mit der Auto-Inkrement-Funktion erstellt)
- die ID des neuen Bilds in der Spalte `image_id` in die Tabelle `article` einfügen

Außerdem kann die Datenbank immer nur einer Tabelle Daten hinzufügen; daher müssen bei jedem Hinzufügen eines neuen Artikels die Tabellen `image` und `article` durch separate SQL-Anweisungen ergänzt werden.

Bei der Ausführung nennt man diese SQL-Anweisungen **Operationen**. Eine Transaktion stellt eine Aufgabe dar, die mehrere Datenbankoperationen umfassen kann.

Indem PDO mehr als eine Operation in eine Transaktion einbezieht, kann es überprüfen, ob alle SQL-Anweisungen in der Transaktion erfolgreich ausgeführt werden:

- Wenn sie keine Ausnahme verursachen, können die Änderungen an die Datenbank übertragen werden.
- Wenn es ein Problem bei der Ausführung einer der Anweisungen gibt, löst PDO eine Ausnahme aus.

Der PHP-Code kann dann PDO anweisen, dass keine der Änderungen in der Datenbank gespeichert wird.

Dies wird als **Rollback** der Transaktion bezeichnet.

Transaktionen werden mithilfe von try- und catch-Blöcken durchgeführt.

- Der try-Block enthält den Code, der ausgeführt werden soll.
- Mit dem catch-Block wird eine Ausnahme behandelt, falls eine solche im try-Block ausgelöst wird.

Die Anweisungen innerhalb des try-Blocks weisen PDO an:

- eine Transaktion zu starten
- alle einzelnen SQL-Anweisungen auszuführen, die die Transaktion bilden
- die Änderungen in der Datenbank festzuhalten

Wenn die Ausführung einer der Anweisungen zu einer Ausnahme führt, führt der PHP-Interpreter sofort den Code aus, der sich im nachfolgenden catch-Block befindet.

Der Catch-Block muss:

- PDO verwenden, um sicherzustellen, dass die Datenbank alle vorgenommenen Änderungen rückgängig macht, sodass sie die gleichen Daten enthält wie vor dem Start der Transaktion.
- die Ausnahme erneut auslösen, damit sie von der Standardfunktion zur Ausnahmebehandlung abgefangen werden kann.

Dadurch wird gewährleistet, dass entweder alle SQL-Anweisungen ausgeführt werden oder, falls im try-Block eine Ausnahme ausgelöst wird, keine der Änderungen gespeichert wird.

Das PDO-Objekt verfügt über drei Methoden, die es Ihnen ermöglichen, eine Transaktion zu starten, die Änderungen an die Datenbank zu übertragen oder die Änderungen rückgängig zu machen, sodass die Datenbank dieselben Daten enthält wie vor den Änderungen.

Transaktionen verwenden die drei in der Tabelle rechts aufgeführten Methoden des PDO-Objekts.

Die ersten beiden werden im `try`-Block verwendet; die dritte wird im `catch`-Block verwendet.

1. Ein `try`-Block enthält den Code zur Ausführung aller SQL-Anweisungen in der Transaktion.
2. Die erste Anweisung ruft die Methode `beginTransaction()` des PDO-Objekts auf, um die Transaktion zu starten.
3. Danach folgt der Code zur Ausführung der an der Transaktion beteiligten SQL-Anweisungen.
4. Die letzte Anweisung im `try`-Block ruft die `commit()`-Methode des PDO-Objekts auf, um die Änderungen zu speichern.

METHODE	BESCHREIBUNG
<code>beginTransaction()</code>	Start einer Transaktion
<code>commit()</code>	Änderungen in der Datenbank speichern
<code>rollBack()</code>	Änderungen in der Transaktion rückgängig machen

5. Wenn eine Ausnahme ausgelöst wird, während eine der SQL-Anweisungen ausgeführt wird, wird der Code im `try`-Block angehalten, und der nachfolgende `catch`-Block behandelt die Ausnahme.
6. Die `rollBack()`-Methode des PDO-Objekts wird aufgerufen, um sicherzustellen, dass keine der Änderungen im `try`-Block in der Datenbank gespeichert werden.
7. Das Ausnahmeobjekt wird erneut ausgelöst, damit es von der Standard-Ausnahmebehandlungsfunktion abgegriffen werden kann.

```
① try {  
②     $pdo->beginTransaction();  
③     // SQL-Statements hier ausführen  
④     $pdo->commit();  
⑤ } catch (PDOException $e) {  
⑥     $pdo->rollBack();  
⑦     throw $e;  
}
```

ARTIKEL: SEITE EINRICHTEN (TEIL A)

Die Seite `article.php` verwendet einen ähnlichen Kontrollfluss wie `category.php`. Wenn der Abfragestring

- den Namen `id` hat, sollte sein Wert die ID eines Artikels sein, den der Benutzer zu bearbeiten versucht.
- nicht den Namen `id` hat, wird die Seite zum Erstellen eines neuen Artikels verwendet.

1. Die strikte Typisierung wird deklariert, und erforderliche Dateien werden eingefügt.

2. Der Pfad des Upload-Ordners wird in `$uploads` (S. 306) gespeichert, erlaubte Bildtypen in `$file_types`, erlaubte Dateierweiterungen in `$file_exts` und die maximale Dateigröße in `$max_size`.

3. Die Funktion `filter_input()` prüft auf den Namen `id` im Abfragestring. Enthält er eine gültige Ganzzahl, wird er in `$id` gespeichert. Enthält er einen ungültigen Wert, wird `$id` als `false` gespeichert. Ist er nicht vorhanden, wird `$id` als `null` gespeichert.

4. Um zu prüfen, ob eine Datei hochgeladen wurde, versucht die Seite, den temporären Speicherort der Datei zu ermitteln und in `$temp` zu speichern. Mit dem Null-Koaleszenz-Operator wird ein leerer String in `$temp` gespeichert, wenn ein Bild nicht hochgeladen wurde.

5. Die Variable `$destination` wird initialisiert. Wenn ein Bild hochgeladen wurde, wird sie aktualisiert, um den Pfad zu speichern, in dem das Bild gespeichert werden soll.

6. Das Array `$article` wird deklariert und mit Standardwerten initialisiert, sodass im Formular in Teil D etwas angezeigt wird, wenn ein Benutzer im Begriff ist, einen neuen Artikel zu erstellen (aber noch keine Werte eingegeben hat).

Damit der Beispielcode auf die rechte Seite passt, werden in jeder Codezeile zwei Elemente des Arrays deklariert; im Code-Download steht jedes Element in einer neuen Zeile.

7. Das Array `$errors` wird mit leeren Zeichenketten initialisiert. Die Werte in diesem Array werden nach jedem Formularsteuerelement angezeigt.

8. Eine `if`-Anweisung prüft, ob im Abfragestring eine `ID` angegeben wurde. Ist dies der Fall, bearbeitet die Seite einen Artikel, und die Artikeldaten müssen aus der Datenbank abgefragt werden.

9. `$sql` speichert die SQL-Abfrage zum Abrufen der Artikeldaten.

10. Die Funktion `pdo()` führt die SQL-Anweisung aus, und die `fetch()`-Methode des `PDOStatement`-Objekts ruft die Artikeldaten ab. Die zurückgegebenen Werte überschreiben die Daten, die im in Schritt 6 erstellten Array `$article` gespeichert sind.

Wenn der Artikel nicht gefunden wird, wird `$article` auf `false` gesetzt.

11. Wenn die Variable `$article` den Wert `false` enthält, wird der Benutzer zu `articles.php` mit einer Fehlermeldung weitergeleitet, die besagt, dass der Artikel nicht gefunden werden konnte.

12. Wenn ein Bild für den Artikel gespeichert wurde, besitzt das `image_file`-Element des `$article`-Arrays einen Wert, sodass `$saved_image` den Wert `true` erhält. Ist kein Bild vorhanden, erhält es den Wert `false`.

13. Alle Autoren und Kategorien werden aus der Datenbank gezogen, damit die Drop-down-Felder zur Auswahl des Autors und der Kategorie erstellt und die ausgewählten Werte validiert werden können. Die SQL-Abfrage in `$sql` ruft zunächst die ID, den Vornamen und Nachnamen jedes Mitglieds ab.

14. Die Funktion `pdo()` führt die Abfrage aus, und die Methode `fetchAll()` des `PDOStatement`-Objekts sammelt die Ergebnisse und speichert sie in der Variablen `$authors`. (Es wird ein indiziertes Array zurückgegeben, und der Wert für jedes Element ist ein assoziatives Array mit Mitgliederdaten).

15. Als Nächstes enthält die Variable `$sql` die SQL-Abfrage für die ID und den Namen aller Kategorien.

16. Die Funktion `pdo()` führt die Abfrage aus, und die Methode `fetchAll()` des `PDOStatement`-Objekts ruft die Kategoriedaten ab; sie werden in `$categories` gespeichert.

```

<?php
// Teil A: Einrichtung
declare(strict_types = 1); // Strikte Typisierung
include '../includes/database-connection.php'; // Datenbankverbindung
① include '../includes/functions.php'; // Funktionen
include '../includes/validate.php'; // Funktionen validieren
$uploads = dirname(__DIR__, 1) . DIRECTORY_SEPARATOR . 'uploads' . DIRECTORY_SEPARATOR; // uploads
② $file_types = ['image/jpeg', 'image/png', 'image/gif',]; // Gültige Typen
$file_exts = ['jpg', 'jpeg', 'png', 'gif',]; // Gültige Erweiterungen
$max_size = 5242880; // Max. Dateigröße

// Variablen für den PHP-Code initialisieren
③ $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // id + Validierung
④ $temp = $_FILES['image']['tmp_name'] ?? ''; // Temporäres Bild
⑤ $destination = '';
// Von HTML-Seite benötigte Variablen speichern
$article = [
    'id' => $id, 'title' => '',
    'summary' => '', 'content' => '',
    'member_id' => 0, 'category_id' => 0,
    'image_id' => null, 'published' => false,
    'image_file' => '', 'image_alt' => '',
];
// Artikeldaten
$errors = [
    'warning' => '', 'title' => '', 'summary' => '', 'content' => '',
    'author' => '', 'category' => '', 'image_file' => '', 'image_alt' => '',
];
// Fehlermeldungen

// Wenn id, bearbeitet die Seite einen Artikel -> aktuelle Artikeldaten abrufen
⑧ if ($id) { // Wenn id
    $sql = "SELECT a.id, a.title, a.summary, a.content,
            a.category_id, a.member_id, a.image_id, a.published,
            i.file AS image_file,
            i.alt AS image_alt
        FROM article AS a
        LEFT JOIN image AS i ON a.image_id = i.id
        WHERE a.id = :id;"; // SQL Artikel
    $article = pdo($pdo, $sql, [$id])->fetch(); // Artikeldaten
    if (!$article) { // Kein Artikel
        redirect('articles.php', ['failure' => 'Article not found']); // Redirect
    }
}

⑫ $saved_image = $article['image_file'] ? true : false; // Bild hochgeladen?

// Alle Mitglieder und alle Kategorien
⑬ $sql = "SELECT id, forename, surname FROM member;"; // SQL für alle Mitglieder
⑭ $authors = pdo($pdo, $sql)->fetchAll();
⑮ $sql = "SELECT id, name FROM category;"; // SQL für alle Kategorien
⑯ $categories = pdo($pdo, $sql)->fetchAll();

```

ARTIKEL: DATEN ABRUFEN UND VALIDIEREN (TEIL B)

Diese beiden Seiten zeigen den Code für Teil B, der die vom Benutzer gesendeten Daten sammelt und validiert.

1. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
2. Das Element `image_file` wird dem Array `$errors` hinzugefügt, und ein ternärer Operator weist ihm einen Wert zu. Wenn eine Datei nicht hochgeladen werden konnte, weil sie die maximale Dateigröße in `php.ini` oder `.htaccess` überschreitet, wird ein Fehler gespeichert, andernfalls ein leerer String.
3. Eine `if`-Anweisung prüft, ob eine Datei hochgeladen wurde und keine Fehler aufgetreten sind. Wenn ja, wird die Datei validiert.
4. Da ein Bild hochgeladen wurde, wird sein Alt-Text abgerufen und im Array `$article` gespeichert.
5. Wenn der Medientyp des Bilds ein erlaubter Dateityp ist (festgelegt in Schritt 2 auf der vorherigen Seite), wird ein leerer String zum Wert im `image_file`-Element des Arrays `$errors` hinzugefügt, andernfalls eine Fehlermeldung.
6. Wenn die Dateierweiterung gültig ist (festgelegt in Schritt 2 auf der vorherigen Seite), wird ein leerer String zum Wert im `image_file`-Element des `$errors`-Arrays hinzugefügt, andernfalls eine Fehlermeldung.
7. Überschreitet die Datei die maximale Größe (festgelegt in Schritt 2 auf der vorherigen Seite), wird dem Wert in `image_file` eine Meldung hinzugefügt, die besagt, dass die Datei zu groß ist.
8. Der Schlüssel `image_alt` wird dem Array `$errors` hinzugefügt. Ist der Alt-Text 1-254 Zeichen lang, wird eine leere Zeichenkette gespeichert, andernfalls eine Fehlermeldung.
9. Eine `if`-Anweisung überprüft, ob die Schlüssel `image_file` und `image_alt` des Arrays `$errors` leer sind. Ist dies der Fall, kann das Bild verarbeitet werden.

10. Die Funktion `create_filename()` (die sich in `functions.php` befindet und auf S. 296-297 vorgestellt wurde) entfernt unerwünschte Zeichen aus dem Dateinamen und stellt sicher, dass er eindeutig ist. Der Name wird im Schlüssel `image_file` des Arrays `$article` gespeichert.
11. Die Variable `$destination` enthält den Pfad, in den das Bild hochgeladen werden soll. Dieser wird erstellt, indem der Pfad zum `Uploads`-Ordner (gespeichert in der `$uploads`-Variablen in Schritt 2 auf der vorherigen Seite) mit dem im vorherigen Schritt erstellten Dateinamen verbunden wird.
12. Die Artikeldaten werden aus dem Formular abgerufen. Wenn ein Artikel aktualisiert wird, überschreiben diese Werte die vorhandenen Werte, die in den Schritten 9-10 auf der vorherigen Seite aus der Datenbank gezogen wurden.
13. Die Option zur Veröffentlichung des Artikels ist ein Kontrollfeld. Es wird nur dann an den Server gesendet, wenn es angekreuzt wurde. Der Wert 1 wird zugewiesen, wenn das Kontrollfeld angekreuzt wurde, und 0, wenn nicht: 0 und 1 sind die Werte, die die Datenbank zur Darstellung der booleschen Werte wahr und falsch verwendet.
14. Jedes Textelement wird mit den in Kapitel 6 erstellten Funktionen überprüft. Sind die Daten gültig, speichert das Element eine leere Zeichenfolge. Andernfalls speichert es eine Fehlermeldung für dieses Steuerelement.
15. Die Funktionen `is_member_id()` und `is_category_id()` wurden in die Datei `validate.php` eingefügt. Sie durchlaufen das Array der Mitglieder und Kategorien, die in den Schritten 13-16 auf der vorherigen Seite abgerufen wurden, um zu prüfen, ob der angegebene Wert gültig ist.
16. Die Werte im Array `$errors` werden mit der PHP-Funktion `implode()` zu einer Zeichenkette zusammengefügt und in der Variablen `$invalid` gespeichert. Mit dieser wird festgestellt, ob die Daten in der Datenbank gespeichert werden sollen oder nicht.

```

// Teil B: Formulardaten sammeln und validieren
① if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular gesendet
    // Wenn Datei größer als Limit in php.ini oder .htaccess: Fehlermeldung
    ② $errors['image_file'] = ($_FILES['image']['error'] == 1) ? 'File too big' : '';
}

// Wenn Bild hochgeladen, Bilddaten holen und validieren
③ if ($temp and $_FILES['image']['error'] == 0) { // Wenn Dateihochgeladen
    ④ $article['image_alt'] = $_POST['image_alt']; // Alt-Text
    // Bilddatei validieren
    ⑤ $errors['image_file'] .= in_array(mime_content_type($temp), $file_types)
        ? '' : 'Wrong file type.'; // Dateityp prüfen
    $ext = strtolower(pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION));
    ⑥ $errors['image_file'] .= in_array($ext, $file_extensions)
        ? '' : 'Wrong file extension.'; // Erweiterung prüfen
    ⑦ $errors['image_file'] .= ($_FILES['image']['size'] <= $max_size)
        ? '' : 'File too big.'; // Größe prüfen
    ⑧ $errors['image_alt'] = (is_text($article['image_alt'], 1, 254))
        ? '' : 'Alt text must be 1-254 characters.'; // Alt-Text prüfen
    // Wenn Bilddatei valide, Speicherort festlegen
    ⑨ if ($errors['image_file'] == '' and $errors['image_alt'] == '') {
        ⑩ $article['image_file'] = create_filename($_FILES['image']['name'], $uploads);
        ⑪ $destination = $uploads . $article['image_file'];
    }
}

// Artikeldaten
$article['title'] = $_POST['title']; // Titel
$article['summary'] = $_POST['summary']; // Zusammenfassung
$article['content'] = $_POST['content']; // Inhalt
$article['member_id'] = $_POST['member_id']; // Autor
$article['category_id'] = $_POST['category_id']; // Kategorie
$article['published'] = (isset($_POST['published']) and ($_POST['published'] == 1)) ? 1 : 0; // Veröffentlicht?

// Artikeldaten validieren und Fehlermeldungen, wenn ungültig
$errors['title'] = is_text($article['title'], 1, 80)
    ? '' : 'Title must be 1-80 characters';
$errors['summary'] = is_text($article['summary'], 1, 254)
    ? '' : 'Summary must be 1-254 characters';
$errors['content'] = is_text($article['content'], 1, 100000)
    ? '' : 'Article must be 1-100,000 characters';
$errors['member'] = is_member_id($article['member_id'], $authors)
    ? '' : 'Please select an author';
$errors['category'] = is_category_id($article['category_id'], $categories)
    ? '' : 'Please select a category';
$invalid = implode($errors);

```

ARTIKEL: ÄNDERUNGEN SPEICHERN (TEIL C)

Diese beiden Seiten zeigen den Code für Teil C.

1. Eine if-Anweisung prüft, ob \$invalid Fehlermeldungen enthält. Ist dies der Fall, wird im Array \$errors eine Meldung gespeichert, die den Benutzer auffordert, die Fehler im Formular zu korrigieren.

2. Andernfalls sind die Daten gültig und können verarbeitet werden.

3. Die Daten in \$article werden nach \$arguments kopiert.

Die Funktion pdo() verwendet die Werte in \$arguments. Das HTML-Formular in Teil D verwendet die Werte in \$article.

4. Ein try-Block enthält den Code zur Aktualisierung der Datenbank.

5. Es wird eine Transaktion gestartet, da zwei SQL-Anweisungen zum Erstellen oder Aktualisieren eines Artikels erforderlich sind. Wenn eine fehlschlägt, sollten beide fehlschlagen.

6. Wenn \$destination einen Wert enthält (Schritt 11 auf S. 513), wurde ein Bild hochgeladen. Das Bild muss vor den Artikeldaten verarbeitet werden, da die Tabelle article die ID speichern muss, die für das Bild erstellt wird.

7. Mit Imagick wird die Größe des Bilds geändert und dieses gespeichert:

Ein Imagick-Objekt wird für das hochgeladene Bild erstellt (der Pfad befindet sich in \$temp - Schritt 4 S. 511). Es wird auf 1200 x 700 Pixel verkleinert und unter dem Pfad in \$destination gespeichert.

8. \$sql enthält die SQL-Anweisung, um den Dateinamen und den Alt-Text des Bilds in die Tabelle image der Datenbank einzufügen.

9. Die Funktion pdo() führt die SQL-Anweisung aus. Die Argumente (Bilddatei und Alt-Text) werden als indiziertes Array an die pdo()-Funktion übergeben.

10. Die ID des Bilds wird mit der Methode lastInsertId() des PDO-Objekts erfasst und im Schlüssel image_id des in Schritt 3 erstellten Arrays \$arguments gespeichert.

- 11.** Die Elemente mit den Schlüsseln image_file und image_alt werden aus dem Array \$arguments entfernt, da das Array nur ein Element für jeden Platzhalter in der zweiten SQL-Anweisung enthalten darf.
- 12.** Eine if-Anweisung prüft, ob eine ID angegeben wurde. Wenn ja, wird ein Artikel aktualisiert, und die Variable \$sql enthält eine SQL-Anweisung zur Aktualisierung der Datenbank.
- 13.** Wenn keine ID angegeben wurde, wird ein neuer Artikel erstellt, also wird id aus dem Array \$arguments entfernt, und \$sql enthält die SQL-Anweisung zum Hinzufügen eines neuen Artikels zur Datenbank.
- 14.** Die Funktion pdo() führt die SQL-Anweisung aus.
- 15.** Die commit()-Methode des PDO-Objekts wird aufgerufen, um beide Änderungen in der Transaktion in der Datenbank zu speichern.
- 16.** Wenn der Code noch läuft, wird der Artikel gespeichert, und redirect() schickt den Benutzer zu articles.php.
- 17.** Wenn PDO im try-Block eine Ausnahme auslöst, wird der catch-Block ausgeführt. Das Ausnahmeobjekt wird in \$e gespeichert.
- 18.** Die rollback()-Methode des PDO-Objekts verhindert, dass die Datenbank die Änderungen in der Transaktion speichert.
- 19.** Wenn in Schritt 7 ein Bild auf dem Server gespeichert wurde, wird es mit der PHP-Methode unlink() gelöscht.
- 20.** Wenn der Fehlercode des PDOException-Objekts 1062 ist, ist der Titel in Gebrauch, also wird ein Fehler in \$errors gespeichert.
- 21.** Ansonsten wird die Ausnahme erneut ausgelöst.
- 22.** Wenn die nächste Zeile läuft, muss der Artikel ungültige Daten enthalten haben. Wenn der Artikel bereits ein Bild enthielt (siehe Schritt 12 S. 511), wird es im Array \$article gespeichert. Wenn nicht, wird image_file auf eine leere Zeichenkette gesetzt.

```
// Teil C: Daten auf Validität prüfen, wenn ja -> Datenbank aktualisieren
① if ($invalid) { // Wenn nicht
    $errors['warning'] = 'Please correct the errors below'; // Nachricht
} else { // Sonst
    $arguments = $article; // Artikeldaten speichern
    try {
        $pdo->beginTransaction(); // Start Transaktion
        if ($destination) { // Wenn Bild valide
            $imagick = new \Imagick($temp); // Imagick-Objekt erstellen
            $imagick->cropThumbnailImage(1200, 700); // Bildzuschnitt
            $imagick->writeImage($destination); // Datei speichern
            $sql = "INSERT INTO image (file, alt) // SQL Bild hinzufügen
VALUES (:file, :alt);";
            pdo($pdo, $sql, [$arguments['image_file'], $arguments['image_alt']]);
            $arguments['image_id'] = $pdo->lastInsertId(); // Neue Bild-id
        }
        unset($arguments['image_file'], $arguments['image_alt']);
        if ($id) {
            $sql = "UPDATE article
SET title = :title, summary = :summary, content = :content,
category_id = :category_id, member_id = :member_id,
image_id = :image_id, published = :published
WHERE id = :id;"; // SQL Artikel aktualisieren
        } else {
            unset($arguments['id']); // id entfernen
            $sql = "INSERT INTO article (title, summary, content, category_id,
member_id, image_id, published)
VALUES (:title, :summary, :content, :category_id, :member_id,
:image_id, :published)"; // SQL Artikel erstellen
        }
        pdo($pdo, $sql, $arguments); // SQL Artikel hinzufügen
        $pdo->commit();
        redirect('articles.php', ['success' => 'Article saved']); // Redirect
    } catch (PDOException $e) { // Wenn PDOException
        $pdo->rollBack(); // SQL-Änderungen zurück
        if (file_exists($destination)) { // Wenn Bilddatei vorhanden
            unlink($destination); // Bilddatei löschen
        } // Wenn die Ausnahme PDO-Exception und Integritätsbedingung
        if ($e->errorInfo[1] == 1062) {
            $errors['warning'] = 'Article title already used'; // Warnung
        } else { // Sonst
            throw $e; // Ausnahme werfen
        }
    }
} // neues Bild hochgeladen, Daten nicht gültig -> Bild aus $article entfernen
$article['image_file'] = $saved_image ? $article['image_file'] : '';
```

ARTIKEL: FORMULAR/ MELDUNG (TEIL D)

Dem Benutzer wird immer das gleiche Formular angezeigt, unabhängig davon, ob er einen Artikel erstellt oder bearbeitet.

HINWEIS: Das Formular im Code-Download enthält weitere HTML-Elemente und -Attribute für die Formularsteuerelement-Labels und zur Steuerung der Formulardarstellung. Sie wurden aus dem Code auf der rechten Seite entfernt, damit er auf einer Seite Platz findet und Sie sich auf das Wesentliche konzentrieren können.

1. Das Attribut `action` des öffnenden `<form>`-Tags verweist auf die Seite `article.php`. Der Abfragestring enthält den Namen `id`; wenn die Seite einen bestehenden Artikel aktualisiert, dann ist sein Wert die `id` des Artikels (die im Attribut `$id` am Anfang der Datei gespeichert wurde). Die Daten werden mit `HTTP POST` gesendet.

2. Wenn es einen Wert für den Schlüssel `warning` des Arrays `$errors` gibt, wird dieser dem Benutzer angezeigt.

3. Das Formular zum Hochladen eines Bilds wird nur angezeigt, wenn noch *kein* Bild für den Artikel bereitgestellt wurde.

4. Die Besucher erhalten die Möglichkeit, ein Bild hochzuladen.

5. Wenn das Array `$errors` eine Fehlermeldung für dieses Steuerelement enthält, wird diese nach der Dateieingabe angezeigt.

Ein entsprechender Schritt wird nach allen Formularsteuerelementen mit Ausnahme des Kontrollkästchens für die Veröffentlichung durchgeführt.

6. Eine Texteingabe ermöglicht es dem Besucher, einen Alt-Text anzugeben.

7. Wenn ein Bild hochgeladen wurde (und die Daten gültig waren), wird dieses auf der Seite angezeigt, gefolgt von dem Alt-Text.

8. Darunter befinden sich zwei Links: Mit dem ersten kann der Nutzer den Alt-Text bearbeiten, mit dem zweiten das Bild löschen.

9. Der Titel des Artikels wird über eine Texteingabe eingegeben.

Wurde bereits ein Wert angegeben, wird dieser zum Attribut `value` hinzugefügt. Alle reservierten Zeichen werden durch Entitäten ersetzt, um einen XSS-Angriff zu verhindern.

10. Für die Zusammenfassung wird ein `<textarea>`-Element verwendet. Wenn eine Zusammenfassung angegeben wurde, wird sie zwischen den `<textarea>`-Tags ausgeschrieben, wobei die Funktion `html_escape()` verwendet wird, um reservierte Zeichen durch Entities zu ersetzen.

11. Der Hauptinhalt des Artikels verwendet ein weiteres `<textarea>`-Element. Wenn ein Wert angegeben wurde, wird er zwischen den `<textarea>`-Tags ausgeschrieben.

12. Das Autorenauswahlfeld zeigt eine Liste aller Mitglieder an, die den Artikel geschrieben haben könnten.

13. Es wird mit einer `foreach`-Schleife aufgebaut, die das Array aller Mitglieder durchläuft (in Teil A in der Variablen `$authors` gespeichert).

14. Ein `<option>`-Element wird für jedes Mitglied hinzugefügt.

15. Die Bedingung eines ternären Operators prüft, ob ein Autor angegeben wurde und seine ID mit der ID des aktuellen Autors übereinstimmt, der dem Auswahlfeld hinzugefügt wird.

Ist dies der Fall, wird das Attribut `selected` hinzugefügt, um das aktuelle Mitglied als Autor des Artikels zu kennzeichnen.

16. Der Name des Mitglieds wird angezeigt.

17. Mithilfe des in `$categories` gespeicherten `category`-Arrays wird die Auswahlbox für die Kategorien erstellt.

18. Ein Kontrollkästchen gibt an, ob der Artikel veröffentlicht (auf der Website angezeigt) werden soll oder nicht. Ein ternärer Operator prüft, ob die Option angekreuzt wurde; wenn ja, wird dem Element das Attribut `checked` hinzugefügt.

```
<!-- Part D - Display form -->
① <form action="article.php?id=<?= $id ?>" method="post" enctype="multipart/form-data">
    <h2>Edit Articles</h2>
    <?php if ($errors['warning']) { ?>
        <div class="alert alert-danger"><?= $errors['warning'] ?></div>
    <?php } ?>

    <?php if (!$article['image_file']) { ?>
        Upload image: <input type="file" name="image" class="form-control-file" id="image">
        <span class="errors"><?= $errors['image_file'] ?></span>
        Alt text: <input type="text" name="image_alt">
        <span class="errors"><?= $errors['image_alt'] ?></span>
    <?php } else { ?>
        <label>Image:</label> ">
        <p class="alt"><strong>Alt text:</strong> <?= html_escape($article['image_alt']) ?></p>
        <a href="alt-text-edit.php?id=<?= $article['id'] ?>">Edit alt text</a>
        <a href="image-delete.php?id=<?= $id ?>">Delete image</a><br><br>
    <?php } ?>

    <⑨ Title: <input type="text" name="title" value="<?= html_escape($article['title']) ?>">
    <span class="errors"><?= $errors['title'] ?></span>
    <⑩ Summary: <textarea name="summary"><?= html_escape($article['summary']) ?></textarea>
    <span class="errors"><?= $errors['summary'] ?></span>
    <⑪ Content: <textarea name="content"><?= html_escape($article['content']) ?></textarea>
    <span class="errors"><?= $errors['content'] ?></span>
    <⑫ Author: <select name="member_id">
        <?php foreach ($authors as $author) { ?>
            <option value="<?= $author['id'] ?>">
                <?= ($article['author_id'] == $author['id']) ? 'selected' : '' ; ?>>
                <?= html_escape($author['forename'] . ' ' . $author['surname']) ?>
            </option>
        <?php } ?></select>
    <span class="errors"><?= $errors['author'] ?></span>
    <⑯ Category: <select name="category_id">
        <?php foreach ($categories as $category) { ?>
            <option value="<?= $category['id'] ?>">
                <?= ($article['category_id'] == $category['id']) ? 'selected' : '' ; ?>>
                <?= html_escape($category['name']) ?>
            </option>
        <?php } ?></select>
    <span class="errors"><?= $errors['category'] ?></span>
    <input type="checkbox" name="published" value="1"
        <?= ($article['published'] == 1) ? 'checked' : '' ?>> Published
    <input type="submit" name="create" value="save" class="btn btn-primary">
</form>
```

EINEN ARTIKEL LÖSCHEN

Die Seite zum Löschen eines Artikels entspricht der Seite zum Löschen einer Kategorie, mit einem Bestätigungsformular.

1. Die Seite deklariert die strikte Typisierung, und die erforderlichen Dateien werden eingefügt.
2. Die PHP-Funktion `filter_input()` sucht im Abfragestring nach dem Namen `id`. Enthält er eine gültige Ganzzahl, wird er in `$id` gespeichert. Enthält er einen ungültigen Wert, wird `$id` als `false` gespeichert. Ist er nicht vorhanden, enthält `$id` den Wert `null`.
3. Wurde keine `id` gefunden, wird der Benutzer mit einer Fehlermeldung zu `articles.php` weitergeleitet.
4. `$article` wird mit dem Wert `false` initialisiert.
5. `$sql` speichert den SQL-Code für den Artikeltitel, die Bilddatei und die Bildkennung.
6. Die Funktion `pdo()` führt den SQL-Code aus, die Daten über den Artikel werden abgerufen und in `$article` gespeichert.
7. Wurden die Artikeldaten nicht gefunden, wird der Benutzer mit einer Fehlermeldung zu `articles.php` weitergeleitet.
8. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde (um zu bestätigen, dass der Artikel gelöscht werden soll).
9. Wurde das Formular abgeschickt, enthält ein `try`-Block den Code zum Löschen des Artikels.
10. Es wird eine Transaktion gestartet, da das Löschen des Artikel drei SQL-Anweisungen umfassen kann.
11. Eine `if`-Anweisung prüft, ob der Artikel ein Bild enthält.
12. Wenn ja, enthält `$sql` die SQL-Anweisung, um die Spalte `image_id` in der Tabelle `article` für diesen Artikel auf `null` zu setzen, und die Funktion `pdo()` führt die Anweisung aus.

13. `$sql` enthält dann die SQL-Anweisung zum Löschen des Bilds aus der Tabelle `image`, und die Funktion `pdo()` führt diese Anweisung aus.
14. Die Variable `$path` speichert den Pfad zu dem Bild.
15. Eine `if`-Anweisung verwendet die PHP-Funktion `file_exists()`, um zu prüfen, ob die Datei gefunden werden kann. Ist dies der Fall, löscht die PHP-Funktion `unlink()` die Datei (siehe S. 228).
16. Die Variable `$sql` speichert die SQL-Anweisung zum Löschen des Artikels aus der Tabelle `article`, und die Funktion `pdo()` führt die Anweisung aus.
17. Wenn im `try`-Block keine Ausnahme ausgelöst wurde, wird die `commit()`-Funktion des PDO-Objekts aufgerufen, um alle durch die SQL-Anweisungen vorgenommenen Änderungen zu speichern.
18. Der Benutzer wird zu `articles.php` weitergeleitet, mit einer Erfolgsmeldung, dass der Artikel gelöscht wurde.
19. Wenn beim Löschen der Daten eine Ausnahme ausgelöst wurde, wird der Catch-Block ausgeführt.
20. Die Funktion `rollback()` des PDO-Objekts verhindert, dass Änderungen an den SQL-Anweisungen gespeichert werden.
21. Die Ausnahme wird erneut ausgelöst, damit sie von der Standardfunktion zur Ausnahmebehandlung abgefangen werden kann.
22. Wenn die Seite zum ersten Mal geladen wird, zeigt das Formular den Titel des Artikels und eine Submit-Schaltfläche an, um zu bestätigen, dass er gelöscht werden soll. Das Attribut `action` des `<form>`-Tags verwendet die ID des Artikels im Abfragestring.

Probieren Sie es: Erstellen Sie eine Seite, um ein Bild aus einem Artikel zu löschen, und nutzen Sie dabei den gleichen Ansatz wie in dieser Datei gezeigt.

Dann erstellen Sie eine Seite zum Bearbeiten des Alt-Texts.

Die Lösungen für beide Aufgaben sind im Code-Download enthalten.

```

<?php
declare(strict_types = 1); // Strikte Typisierung
① require_once '../includes/database-connection.php'; // Datenbankverbindung
require_once '../includes/functions.php'; // Funktionen
② $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // id validieren
if (!$id) { // Keine valide id
    redirect('articles.php', ['failure' => 'Article not found']); // Redirect mit Error
}
③ $article = false; // $article initialisieren
④ $sql = "SELECT a.title, a.image_id, i.file AS image_file FROM article AS a
    LEFT JOIN image AS i ON a.image_id = i.id WHERE a.id = :id;"; // SQL
⑤ $article = pdo($pdo, $sql, [$id])->fetch(); // Artikeldaten
if (!$article) { // Wenn $article leer
    redirect('articles.php', ['failure' => 'Article not found']); // Redirect
}
⑥ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular gesandt
try {
    $pdo->beginTransaction(); // Start Transaktion
    if ($image_id) { // Wenn Bild
        $sql = "UPDATE article SET image_id = null WHERE id = :article_id;"; // SQL
        pdo($pdo, $sql, [$id]); // Bild aus Artikel löschen
        $sql = "DELETE FROM image WHERE id = :id;"; // SQL Bild löschen
        pdo($pdo, $sql, [$article['image_id']]);
        $path = '../uploads/' . $article['image_file']; // Bildpfad setzen
        if (file_exists($path)) { // Wenn Bilddateivorhanden
            $unlink = unlink($path); // Bilddatei löschen
        }
    }
    $sql = "DELETE FROM article WHERE id = :id;"; // SQL Artikel löschen
    pdo($pdo, $sql, [$id]); // Artikel löschen
    $pdo->commit(); // Transaktion committen
    redirect('articles.php', ['success' => 'Article deleted']); // Redirect + Erfolg
} catch (PDOException $e) { // Wenn Ausnahme
    $pdo->rollBack(); // SQL-Änderungen zurück
    throw $e; // Ausnahme erneut werfen
}
}
?>
<?php include '../includes/admin-header.php' ?> ...
<h2>Delete Article</h2>
<form action="article-delete.php?id=<?= $id ?>" method="POST" class="narrow">
    <p>Click confirm to delete: <i><?= html_escape($article['title']) ?></i></p>
    <input type="submit" name="delete" value="Confirm" class="btn btn-primary">
    <a href="articles.php" class="btn btn-danger">Cancel</a>
</form> ...
<?php include '../includes/admin-footer.php'; ?>

```

ZUSAMMENFASSUNG

DATEN IN DER DATENBANK AKTUALISIEREN

- Benutzerdaten müssen abgerufen und validiert werden, bevor sie der Datenbank hinzugefügt werden.
- Mit der Methode `execute()` des `PDOStatement`-Objekts können SQL-Anweisungen ausgeführt werden, die Daten erstellen, aktualisieren oder löschen.
- SQL kann neue Daten immer nur in eine einzelne Tabelle einfügen.
- Die Methode `getLastInsertId()` des `PDO`-Objekts gibt die ID einer neuen Zeile zurück, wenn sie der Datenbank hinzugefügt wird.
- Die Methode `rowCount()` des `PDOStatement`-Objekts gibt zurück, wie viele Tabellenzeilen betroffen sind, wenn ein SQL `INSERT`-, `UPDATE`- oder `DELETE`-Befehl ausgeführt wird.
- Eine Transaktion führt eine Reihe von SQL-Anweisungen aus und speichert die Änderungen nur, wenn alle Anweisungen fehlerfrei ausgeführt werden.
- Wenn eine SQL-Anweisung eine Eindeutigkeitsbedingung verletzt, wird ein `PDOException`-Objekt ausgelöst. Es enthält einen Fehlercode, der die Ursache der Ausnahme beschreibt.

D

DIE BEISPIEL-
ANWENDUNG
AUSBAUEN

Dieser letzte Abschnitt zeigt die Implementierung von Funktionen, die von vielen Websites verwendet werden. Dabei lernen Sie, wie Sie einer Website neue Funktionen und einige fortgeschrittene Techniken hinzufügen können.

Wenn Sie fünf PHP-Entwickler bitten, die gleiche Website zu erstellen, dann bekommen Sie wahrscheinlich fünf verschiedene Lösungen. Das liegt daran, dass es nicht den einen richtigen Weg zur Erstellung einer Website gibt, sondern viele Möglichkeiten, den Code für die einzelnen Aufgaben zu strukturieren. Trotzdem gibt es einige bewährte Verfahren, von denen Sie lernen können, und die gestalterischen Überlegungen in diesem Abschnitt werden Sie bei der Erstellung Ihrer eigenen Projekte leiten.

Kapitel 14 zeigt Ihnen, wie Sie die Beispelseite in Teilen umbauen können, um Klassen besser zu nutzen. Erfahrene PHP-Entwickler machen oft ausgiebig Gebrauch von benutzerdefinierten Klassen, um den Code zusammenzufassen, der eine Reihe miteinander verwandter Aufgaben ausführt.

In Kapitel 15 sehen Sie, wie Sie von anderen Programmierern für bestimmte Aufgaben geschriebene und der PHP-Community zur Verfügung gestellte PHP-Klassen finden und verwenden können. Dadurch ersparen Sie es sich, den Code zur Ausführung der gleichen Aufgaben selbst zu schreiben.

Kapitel 16 erläutert die Mitgliederregistrierung auf der Website. Mitglieder können sich anmelden, auf sie zugeschnittene Seiten anzeigen und eigene Beiträge erstellen. Sie erfahren auch, wie Sie den Zugang zu den Administrationsseiten einschränken können, damit nur berechtigte Personen diese nutzen können.

In Kapitel 17 erfahren Sie, wie Sie die URLs für die Website besser lesbar und für Suchmaschinen leichter zu indizieren machen. Außerdem lernen Sie, wie Sie den Website-Mitgliedern erlauben, Artikel zu kommentieren und anzugeben, welche Artikel ihnen gefallen.

Jedes der verbleibenden Kapitel bringt eine neue Version der Beispiel-Website mit. Bevor Sie weiterlesen, müssen Sie verstehen, wie diese Dateien strukturiert sind, und einige wichtige Begriffe kennenlernen.

ABSOLUTE & RELATIVE PFADE

Zunächst erfahren Sie den Unterschied zwischen absoluten und relativen Pfaden und wie Sie diese jeweils verwenden sollten.

WIE DATEIEN ORGANISIERT SIND

Als Nächstes lernen Sie, wie die Dateien der Beispielseite reorganisiert wurden. Die vom Browser angeforderten Dateien werden alle in einem Ordner gespeichert, der als Dokumentstammverzeichnis bezeichnet wird. Die anderen, ergänzenden PHP-Dateien (z.B. Klassen) werden jedoch in einem Ordner oberhalb des Dokumentstammverzeichnisses gespeichert, um die Sicherheit zu erhöhen.

NEUE SETUP-DATEIEN

In diesem Abschnitt werden zwei neue Dateien vorgestellt, die häufig bei der Erstellung von Websites verwendet werden:

- `config.php` enthält Einstellungen, die sich ändern, wenn eine Website auf einem neuen Server installiert wird
- `bootstrap.php` enthält den Code, den die Site benötigt, um zu laufen; diese Datei wird von jeder Seite der Website eingebunden

WIE VARIABLEN DATEN SPEICHERN

Schließlich ist es hilfreich zu verstehen, wie der PHP-Interpreter die Daten in Variablen speichert. Ein immer wiederkehrendes Thema in den letzten vier Buchkapiteln ist der Einsatz von Klassen zur besseren Code-Strukturierung. Die von den Nutzern angeforderten PHP-Seiten erstellen auf Grundlage dieser Klassen Objekte und rufen dann deren Methoden auf, um die für die Website benötigten Aufgaben auszuführen.

Aus Platzgründen kann in den verbleibenden Kapiteln nicht jede Datei der Beispiel-Website abgedruckt werden. Deshalb ist es hilfreich, wenn Sie den Code beim Durcharbeiten der Beispiele geöffnet haben. So können Sie die Versionen der Dateien in den einzelnen Kapiteln mit den Dateien in den vorangegangenen Kapiteln vergleichen, um zu sehen, wo sie sich geändert haben.

In Kapitel 16 müssen Sie eine neue Version der Datenbank mit zusätzlichen Tabellen und Daten erstellen, die zur Unterstützung der neuen Funktionen der Website erforderlich sind. Die SQL-Datei zum Erstellen der aktualisierten Version der Datenbank ist im Code-Download enthalten, und Sie werden am Anfang von Kapitel 16 daran erinnert, dies zu tun.

ABSOLUTE UND RELATIVE PFADE

Ein absoluter Pfad beschreibt den genauen Speicherort einer Datei auf einem Computer. Ein relativer Pfad beschreibt den Speicherort einer Datei im Verhältnis zu einem anderen Speicherort.

Um zu verstehen, wie die Dateien auf der Beispielseite organisiert sind, ist es hilfreich, einige Begriffe zu klären:

- Ein **Pfad** gibt den Ort einer Datei oder eines Verzeichnisses (auch als Ordner bezeichnet) an.
- Das **Stammverzeichnis** ist der oberste Ordner auf einem Computer.
- Ein **absoluter Pfad** beschreibt den Speicherort einer Datei oder eines Ordners anhand des Pfads vom Stammverzeichnis zu diesem.

In der Abbildung auf der rechten Seite sehen Sie einen Teil des Beispiel-Codes, der auf einem Computer installiert ist.

Auf einem **Mac** oder **Linux**-PC wird das Stammverzeichnis durch einen Schrägstrich dargestellt. Dann trennen weitere Schrägstriche jeden Ordner oder jede Datei ab, sodass der absolute Pfad zu `files.php` wie folgt lautet: `/Users/Jon/phpbook/section_b/c05/files.php`.

Unter **Windows** ist das Stammverzeichnis ein Laufwerkbuchstabe, gefolgt von einem Doppelpunkt und einem Backslash; nachfolgend trennt ein Backslash jeden Ordner oder jede Datei ab, sodass der absolute Pfad zu `files.php` wie folgt lautet:

`C:\phpbook\section_b\c05\files.php`

Absolute Pfade sind präzise, aber sie können:

- recht lang werden und daher viel Tipparbeit erfordern
- sich ändern, wenn Dateien auf einen anderen Computer verschoben werden

Der Ordner, in dem sich die gerade ausgeführte Datei befindet, ist das **aktuelle Arbeitsverzeichnis**. In der Abbildung auf der rechten Seite ist das aktuelle Arbeitsverzeichnis `c05`, wenn `files.php` ausgeführt wird.

Relative Pfade beschreiben den Speicherort einer anderen Datei in Bezug auf das aktuelle Arbeitsverzeichnis.

Der relative Pfad zu einer anderen Datei im selben Ordner besteht nur aus dem Dateinamen. Dieser beschreibt zum Beispiel die relative Position von `index.php` im Ordner `c05`:

`index.php`

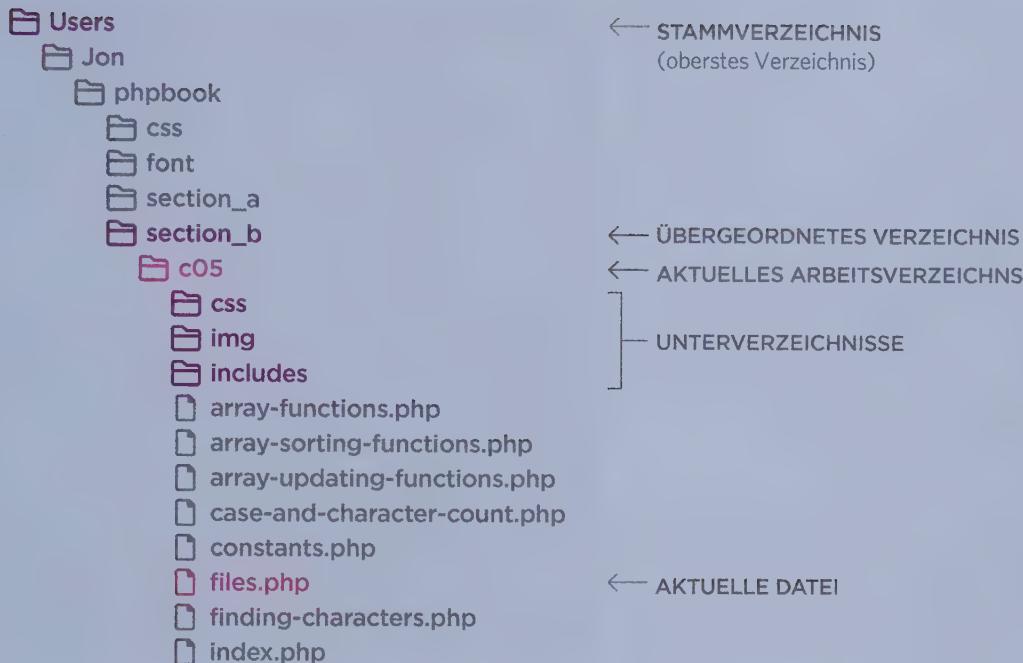
Um den Pfad zu einer in einem Unterordner befindlichen Datei zu beschreiben, verwenden Sie den Namen des Unterordners, gefolgt von einem Schrägstrich und dem Namen der Datei in diesem Verzeichnis. Der relative Pfad zu `header.php` im Ordner `includes` lautet:

`includes/header.php`

Um in der Verzeichnishierarchie einen Schritt nach oben zu gehen, verwenden Sie `..` Der relative Pfad zu der Datei `index.php` im `phpbook`-Verzeichnis wäre:

`../../index.php`

Wie Sie sehen, erfordern relative Pfade weniger Tipparbeit als absolute Pfade. Außerdem funktionieren sie oft auch dann, wenn eine Site auf einen neuen Server umzieht. Wenn die Pfade im Code-Download beispielsweise nur auf Dateien im `phpbook`-Ordner verweisen, ändern sich die relativen Pfade zwischen den Dateien und Ordnern nicht, wenn der Code auf einem anderen Computer ausgeführt wird.



Beim Einbinden einer anderen Datei in eine PHP-Datei ist es sinnvoll, absolute Pfade zu verwenden. Um zu verstehen warum, betrachten Sie folgendes Szenario:

- Eine PHP-Datei im Ordner c05 wird angefordert.
- Sie bindet eine PHP-Datei im Ordner section_a ein.
- Diese Datei bindet eine dritte Datei mit einem relativen Pfad ein.

Der PHP-Interpreter sucht alle drei Dateien relativ zum Ordner c05. Das ist so, als würde der Code aus den eingebundenen Dateien kopiert und in die Datei in c05 eingefügt. In diesem Fall könnte die dritte Datei nicht gefunden werden, da ein relativer Pfad vom Ordner section_a aus nicht dasselbe ist wie ein relativer Pfad vom Ordner c05 aus. Durch die Verwendung von absoluten Pfaden beim Einbinden von Dateien können Sie dies vermeiden.

Da absolute Pfade länger sind (und mehr Tipparbeit erfordern), speichern Websites den ersten Teil eines absoluten Pfads als Konstante. Diese Konstante kann dann zur Erstellung absoluter Pfade genutzt werden.

Das Stammverzeichnis der Anwendung (**application root**) ist der oberste Ordner des Website-Codes. Der absolute Pfad zu *diesem* Ordner wird häufig in der Konstanten gespeichert, die zur Erstellung von Pfaden zu den Include-Dateien verwendet wird. Unten sehen Sie, wie der Pfad zum Anwendungsordner in der Konstanten APP_ROOT gespeichert wird:

- Die PHP-Funktion dirname() (siehe Seite 228) liefert den Pfad zu dem Verzeichnis zurück, das eine Datei enthält.
- Das Argument ist die in PHP integrierte __FILE__-Konstante, die den absoluten Pfad zur aktuellen Datei enthält.

Diese Anweisung speichert also den Pfad zu dem Ordner, in dem sich die aktuelle Datei befindet. Würde sie in einer Datei innerhalb des Ordners c05 verwendet, enthielte die APP_ROOT-Konstante den absoluten Pfad zum Ordner c05. Da der Pfad vom PHP-Interpreter erstellt wird, wäre er auch dann noch korrekt, wenn die Site auf einen neuen Server verschoben werden würde.

```
define('APP_ROOT', dirname(__FILE__));
```

DATEISTRUKTUR UND DOKUMENT- STAMMVERZEICHNIS

Wenn eine Website online geht, müssen sich alle vom Browser abrufbaren Dateien im sogenannten **Dokumentstammverzeichnis** befinden. Der PHP-Interpreter kann jedoch auch auf Dateien *oberhalb* des Dokumentstammverzeichnisses zugreifen.

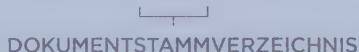
Ein Webserver besitzt noch eine andere Art von Stammverzeichnis, das sogenannte **Dokumentstammverzeichnis** (oder Web-Root). Auf diesen Ordner verweist auch der Domänenname einer Website. Wenn zum Beispiel die URL für die Homepage einer Website wie folgt lautet:

`http://example.org/index.php`

Dann würde der Webserver von example.org im Dokumentstammverzeichnis der Website nach der Datei `index.php` suchen.

Befindet sich die Website auf den Servern eines Hosting-Providers, könnte der absolute Pfad zu dieser Datei etwa wie folgt lauten:

`/var/www/example.org/htdocs/index.php`



Das Dokumentstammverzeichnis kann auf einem Webserver verschiedene Namen haben, trägt aber normalerweise eine Bezeichnung wie `htdocs`, `public`, `public_html`, `web`, `www` oder `wwwroot`.

Jede von einem Browser angeforderte Datei muss im Dokumentstammverzeichnis (oder einem Unterordner davon) gespeichert werden. Dazu gehören vom Benutzer angeforderte Seiten, Bilder oder andere Medien sowie CSS- und JavaScript-Dateien. Der Browser kann keine Dateien oberhalb des Dokumentenstammsatzes anfordern.

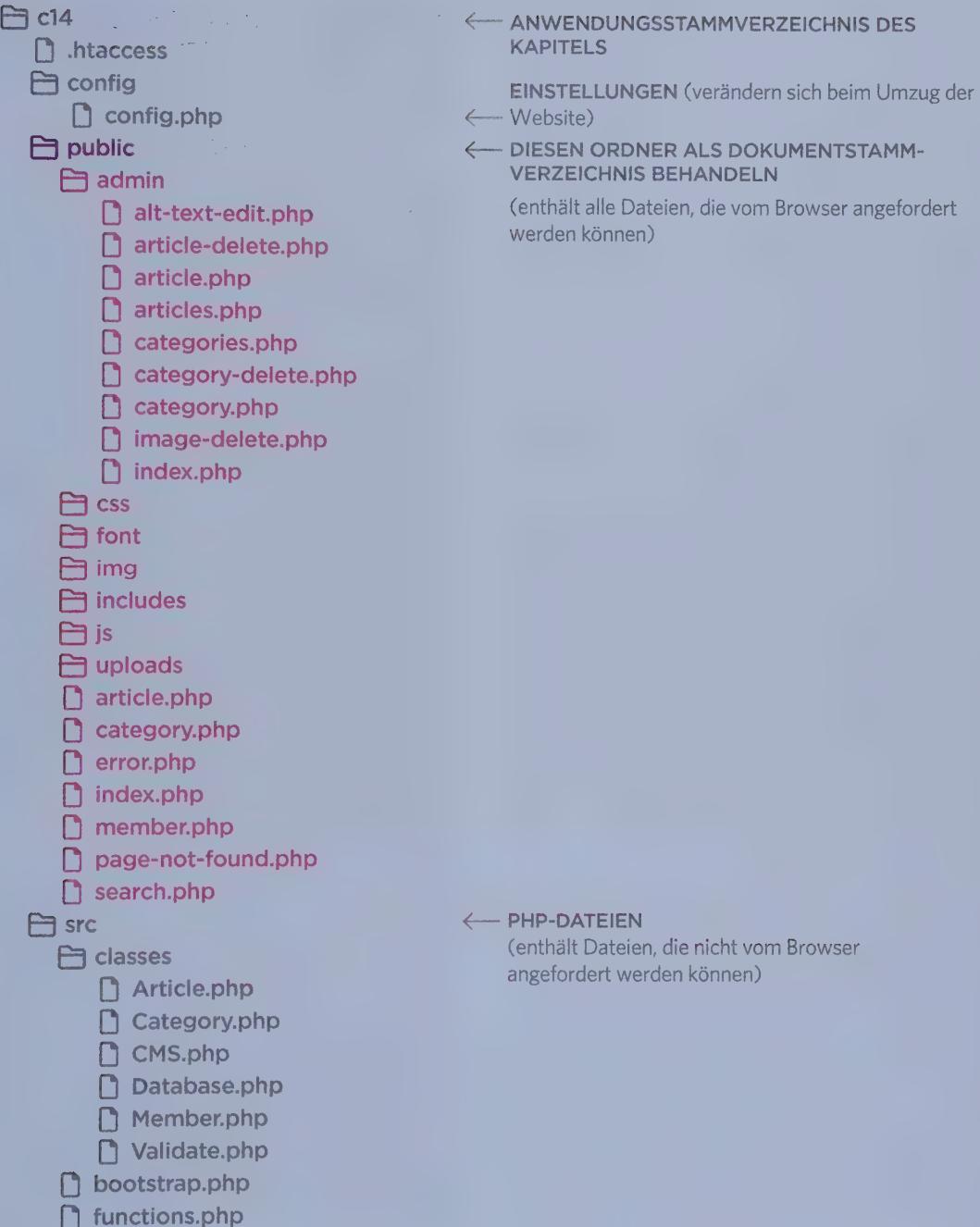
Genau wie Mac- und Linux-Betriebssysteme einen Schrägstrich verwenden, um das Stammverzeichnis der Anwendung (den obersten Ordner auf dem Computer) anzugeben, beziehen sich URLs, die mit einem Schrägstrich beginnen, auf das Dokumentstammverzeichnis; die oberste Dateiebene, auf die der Browser zugreifen kann. Aus diesem Grund beginnen HTML-Links oft mit einem Schrägstrich.

Der Download-Code enthält mehrere Versionen der Beispiel-Website in verschiedenen Ordnern. Daher müssen Sie sich vorstellen, dass der `public`-Ordner in jedem der verbleibenden Kapitel das Dokumentstammverzeichnis der Beispiel-Website für dieses Kapitel ist, auf das der Domainname der Website verweist. Die Site speichert diesen Pfad in der Konstanten `DOC_ROOT`, damit er bei Bedarf verwendet werden kann (siehe Seite 528). Auf einer Live-Website könnte der Code stattdessen einfach einen Schrägstrich verwenden. Der Browser kann zwar nur Dateien im Dokumentstammverzeichnis anfordern, der PHP-Interpreter hat allerdings auch Zugriff auf Dateien oberhalb des Dokumentstammsverzeichnisses. In den nachfolgenden Kapiteln:

- befinden sich die Dateien, die der Besucher über eine URL anfordert, im Dokumentstammverzeichnis (`public`-Ordner) des jeweiligen Kapitels.
- befinden sich sämtliche Dateien, die der Benutzer nicht anfordern kann (z. B. Funktionsdateien und Klassendefinitionen), in Ordnern oberhalb des Dokumentstammverzeichnisses. Dies erhöht die Sicherheit, da es verhindert, dass Benutzer Zugriff auf diese Dateien bekommen.

In Kapitel 14 wäre der Ordner `c14` das Äquivalent zum **Anwendungsstammverzeichnis** (dem Stammsort für die Anwendung). Es gibt zwei neue Ordner, die sich innerhalb des Anwendungsstammverzeichnisses, aber oberhalb des Dokumentstammverzeichnisses befinden:

- `config` enthält eine Datei, in der alle Einstellungen gespeichert werden, die sich ändern können, wenn die Website auf einen neuen Server migriert wird.
- `src` enthält die Funktionen und die Datei `bootstrap.php`. Hier gibt es auch noch einen Unterordner `classes` mit der Klassendefinitionen.



Legende:

- Dokumentstammverzeichnis
- Innerhalb Dokumentstammverzeichnis
- Oberhalb Dokumentstammverzeichnis

CONFIG-DATEI

Wenn eine Website auf einem neuen Server installiert wird, müssen möglicherweise einige Daten aktualisiert werden. Zum Beispiel:

- Verzeichnisnamen/Pfade ändern sich; z.B. kann ein Dokumentstammverzeichnis `htdocs`, `content` oder `public` heißen.
- Datenbankgestützte Websites müssen den Speicherort der im DSN verwendeten Datenbank sowie den Benutzernamen und das Passwort eines Datenbankbenutzerkontos kennen.

Diese Daten werden als **Konfigurationsdaten** bezeichnet, da sie festlegen, wie die Website ausgeführt wird. Sie werden in der Regel als Variablen oder Konstanten in einer Datei hinterlegt. Dies ist daher die einzige Datei, die bei der Installation der Website auf einem neuen Server aktualisiert werden muss. In der Beispielanwendung heißt die Datei `config.php`.

HINWEIS: Sie müssen den Code in dieser Datei für jedes der verbleibenden Kapitel aktualisieren.

1. `DEV` wird auf `true` gesetzt, wenn sich die Website in der Entwicklung befindet, und auf `false`, wenn sie in Betrieb ist. Dies steuert die Art der Fehlerbehandlung.
2. `DOC_ROOT` entspricht dem Pfad zum Dokumentstammverzeichnis (siehe Seite 526).
3. `ROOT_FOLDER` enthält den Namen des Dokumentstammverzeichnisses (z.B. `public`, `content`, oder `htdocs`).
4. Die Einstellungen für die Datenbankverbindung werden in Variablen gespeichert, und dann wird das DSN erstellt.
5. `MEDIA_TYPES` enthält die zulässigen Dateitypen. `FILE_EXTENSIONS` enthält die zulässigen Dateinamenerweiterungen. `MAX_SIZE` ist die maximale Dateigröße (in Kilobytes). `UPLOADS` ist der absolute Pfad zum Ordner `uploads`.

section_d/c14/src/config.php

PHP

```
<?php  
① define('DEV', true);      // In Entwicklung befindlich oder live? In Entwicklung = true | Live = false  
② define("DOC_ROOT", '/phpbook/section_d/c14/public/');    // Pfad vom Dokumentstamm zur Website  
③ define("ROOT_FOLDER", 'public');                         // Name des Dokumentstammverzeichnisses  
// Datenbankeinstellungen  
④ [ $type      = 'mysql';                                // Datenbanktyp  
  $server    = 'localhost';                               // Server, auf dem sich die Datenbank befindet  
  $db        = 'phpbook-1';                               // Name der Datenbank  
  $port      = '';                                     // Port ist in MAMP meist 8889 und in XAMPP 3306  
  $charset   = 'utf8mb4';                                // UTF-8-Kodierung mit 4 Bytes pro Zeichen  
  $username  = 'ENTER YOUR USERNAME';                  // Geben Sie hier IHREN Benutzernamen ein  
  $password  = 'ENTER YOUR PASSWORD';                  // Geben Sie hier IHR Passwort ein  
  $dsn       = "$type:host=$server;dbname=$db;port=$port;charset=$charset"; // NICHT VERÄNDERN  
// Datei-Upload-Einstellungen  
⑤ [ define('MEDIA_TYPES', ['image/jpeg', 'image/png', 'image/gif',]); // Zulässige Dateitypen  
  define('FILE_EXTENSIONS', ['jpeg', 'jpg', 'png', 'gif',]);          // Zulässige Dateinamenerweiterungen  
  define('MAX_SIZE', '5242880');                                // Maximale Dateigröße  
  define('UPLOADS', dirname(__DIR__, 1) . DIRECTORY_SEPARATOR . ROOT_FOLDER .  
          DIRECTORY_SEPARATOR . 'uploads' . DIRECTORY_SEPARATOR); // NICHT VERÄNDERN
```

BOOTSTRAP-DATEI

In Kapitel 13 enthielt jede Seite der Website mehrere Dateien, von denen eine ein PDO-Objekt erstellte. Um diesen Code nicht auf jeder Seite zu wiederholen, beginnt nun jede Seite mit der folgenden Datei, die ihrerseits:

- die Dateien config.php und functions.php einbindet
- Funktionen zur Fehler- und Ausnahmebehandlung und eine neue Funktion zum Laden von Klassendefinitionen festlegt
- ein CMS-Objekt erstellt, das alle Seiten für den Zugriff auf die Datenbank nutzen werden

Wenn eine Datei andere Dateien lädt und Objekte erstellt, die eine Website zum Laufen braucht, dann bekommt sie oft einen Namen wie bootstrap.php oder setup.php.

Sobald das CMS-Objekt erstellt wurde, werden die Datenbankverbindungsdaten gelöscht, damit sie nicht an anderer Stelle auf der Website verwendet (oder versehentlich angezeigt) werden können.

1. Der Pfad zum Anwendungsstammverzeichnis (zwei Verzeichnisebenen oberhalb dieser Datei) wird in der Konstanten APP_ROOT gespeichert.
2. functions.php und config.php werden eingebunden.
3. Die PHP-Funktion spl_autoload_register() (die Sie in Kapitel 14 kennengelernt haben) stellt sicher, dass die Klassendefinitionen nur geladen werden, wenn diese Seite sie benötigt.
4. Eine if-Anweisung überprüft, ob sich die Seite nicht in der Entwicklung befindet (sie ist live). In diesem Fall werden die Standardfunktionen für Ausnahmen, Fehlerbehandlung und Beendigung festgelegt.
5. Ein CMS-Objekt wird erstellt und in der Variablen \$cms gespeichert. Es wird für die Interaktion mit der Datenbank verwendet.
6. Die PHP-Funktion unset() entfernt Variablen mit Datenbankverbindungsdaten, damit sie nicht wieder verwendet werden können.

PHP

```
<?php  
① define('APP_ROOT', dirname(__FILE__, 2));  
② [ require APP_ROOT . '/resources/functions.php';  
② [ require APP_ROOT . '/resources/config.php';  
  
③ [ spl_autoload_register(function($class)  
{  
    $path = APP_ROOT . '/src/classes/';  
    require $path . $class . '.php';  
});  
④ [ if (DEV !== true) {  
    set_exception_handler('handle_exception');  
    set_error_handler('handle_error');  
    register_shutdown_function('handle_shutdown');  
}  
⑤ $cms = new CMS($dsn, $username, $password);  
⑥ unset($dsn, $username, $password);
```

section_d/c14/src/bootstrap.php

```
// Anwendungsstammverzeichnis  
// Funktionen  
// Konfigurationsdaten  
  
// Autoload-Funktion festlegen  
  
// Pfad zu Klassendefinitionen  
// Klassendefinitionen einbinden  
  
// Exception-Handler festlegen  
// Error-Handler festlegen  
// Shutdown-Handler festlegen  
  
// CMS-Objekt erstellen  
// Datenbankverbindungsdaten entfernen
```

WIE VARIABLEN DATEN SPEICHERN

Beim Erstellen einer Variablen speichert der PHP-Interpreter ihren Namen und ihren Wert getrennt. So kann er den von ihm verwendeten Speicher effizienter verwalten. Stellen Sie sich zum besseren Verständnis vor, dass die Werte in einer Reihe von Schließfächern verwahrt werden ...

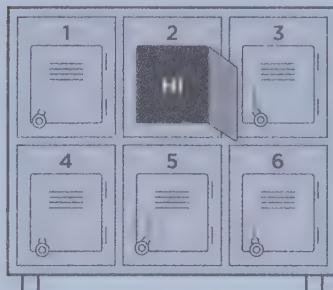
Wenn eine Variable erstellt wird, wird ihr Name zusammen mit einer Schließfachnummer in einer **Symboltabelle** gespeichert. Das entsprechende Schließfach enthält den Wert, für den die Variable steht. Zum Beispiel:

1. \$greeting wird ein Wert zugewiesen
2. \$welcome wird dann der Wert zugewiesen, den \$greeting repräsentiert

```
$greeting = 'Hi';  
$welcome = $greeting;
```

Die Symboltabelle enthält nun zwei Variablennamen. Beide verweisen auf das gleiche Schließfach (oder die gleiche Speicheradresse).

VARIABLE	SPEICHERADRESSE
\$greeting	2
\$welcome	2



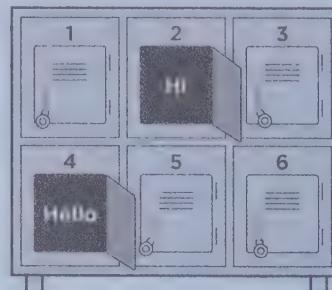
Wenn einer dieser Variablen ein neuer Wert zugewiesen wird, legt der PHP-Interpreter den neuen Wert an einer neuen Speicheradresse ab. Zum Beispiel:

1. \$greeting wird ein Wert zugewiesen
2. \$welcome wird der Wert zugewiesen, den \$greeting repräsentiert
3. \$greeting wird mit einem neuen Wert aktualisiert; dem Text 'Hello'

```
$greeting = 'Hi';  
$welcome = $greeting;  
$greeting = 'Hello';
```

Hier enthält die Symboltabelle immer noch zwei Variablennamen, aber jeder hat seine eigene Speicheradresse.

VARIABLE	SPEICHERADRESSE
\$greeting	4
\$welcome	2

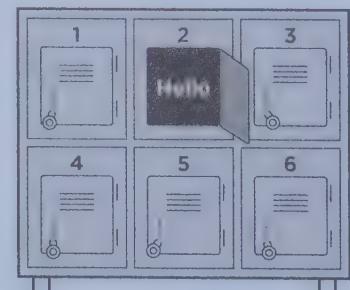


Sie können dem PHP-Interpreter mitteilen, dass der Wert einer Variablen dieselbe Speicheradresse wie eine bestehende Variable verwenden soll. Wenn dann eine der beiden Variablen aktualisiert wird, wird die gemeinsam genutzte Speicheradresse aktualisiert. Dazu setzen Sie ein kaufmännisches Und vor den Variablennamen. Dies wird als **Zuweisung durch Referenz** bezeichnet. Zum Beispiel:

```
$greeting = 'Hi';  
$welcome = &$greeting;  
$greeting = 'Hello';
```

Die Symboltabelle enthält nun zwei Variablennamen, die beide auf dieselbe Speicheradresse verweisen.

VARIABLE	SPEICHERADRESSE
\$greeting	2
\$welcome	2



Funktionsparameter können Verweise auf Werte verwenden, die für global deklarierte Variablen erstellt wurden, und Objekte verhalten sich immer so, als ob sie per Referenz zugewiesen oder übergeben werden.

Beim Ausführen einer Funktion wird eine neue Symboltabelle erstellt, die die in der Funktion deklarierten Parameter- und Variablennamen enthält.

In einer Funktionsdefinition kann festgelegt werden, dass ein Parameter eine Speicheradresse verwenden soll, die für eine globale Variable erstellt wurde. Dadurch kann der Code in der Funktion auf den in dieser globalen Variablen gespeicherten Wert zugreifen und ihn auch aktualisieren.

Dies wird als **Übergabe per Referenz** bezeichnet, da der Funktion ein Verweis auf die Speicheradresse des Variablenwerts übergeben wird (statt ihr den bloßen, durch die globale Variable repräsentierten Wert zu übergeben).

Zur Übergabe per Referenz wird dem Parameternamen in der Funktionsdefinition ein kaufmännisches Und vorangestellt.

```
$current_count = 0; // Globale Variable

function updateCounter(&$counter)
{
    $counter++;      // Zähler um 1 erhöhen
}
```

Mit jedem Aufruf dieser Funktion wird der Wert der Variablen `$current_count` um 1 erhöht. Beachten Sie, dass das kaufmännische Und nur in der Funktionsdefinition verwendet wird, nicht aber beim Aufruf der Funktion:

```
updateCounter($current_count);
```

Objekte verhalten sich **immer** so, als ob sie per Verweis zugewiesen oder übergeben werden. Wenn nachfolgend zum Beispiel das `DateTime`-Objekt erstellt wird, werden in der Symboltabelle der Name der Variablen `$start` und die Speicheradresse, an der das Objekt gespeichert ist, hinterlegt:

```
$start = new DateTime('2021-01-01');
```

Wenn der Wert der `$end`-Variablen mithilfe des in der `$start`-Variablen gespeicherten Objekts gesetzt wird, verweisen beide Variablen auf dieselbe Speicheradresse, da die Variablen per Referenz zugewiesen werden:

```
$end = $start;
```

Wenn das in `$start` gespeicherte Objekt aktualisiert wird, wird demnach auch der Wert für `$end` aktualisiert, da beide Variablennamen auf dieselbe Speicheradresse verweisen:

```
$from->modify('+3 month');
```

Wenn ein Objekt als Argument einer Funktion oder einer Methode verwendet wird, verhält sich das Objekt außerdem so, als wäre es per Referenz übergeben worden (und zwar auch ohne ein kaufmännisches Und). Es ist wichtig, diesen Punkt zu verstehen, da sich im nächsten Kapitel mehrere Objekte einen Verweis auf dasselbe PDO-Objekt teilen.

Im Code-Download befindet sich im Einführungsordner zu diesem Abschnitt eine Datei zur Veranschaulichung der auf diesen beiden Seiten aufgeführten Beispiele.

KAPITEL IN TEIL D

DIE BEISPIELANWENDUNG AUSBAUEN

14

REFACTORING & DEPENDENCY INJECTION

Je stärker eine Website wächst, desto wichtiger ist es, ihren Code sorgfältig zu strukturieren. In diesem Kapitel untersuchen wir, wie sich die Struktur des Codes mit benutzerdefinierten Klassen verbessern lässt, sodass er leichter zu verstehen, zu pflegen und um neue Funktionen zu erweitern ist.

15

NAMENSRÄUME & BIBLIOTHEKEN

Programmierer veröffentlichen oft Code, den sie für bestimmte Aufgaben geschrieben haben, in Bibliotheken und Paketen. Sie können diesen Code in Ihre Website einbinden, statt die Funktionalität für dieselben Aufgaben von Grund auf neu zu schreiben.

16

MITGLIEDSCHAFT

Dieses Kapitel zeigt, wie sich Besucher als Mitglieder der Website registrieren können. Ihre Informationen werden in der Datenbank gespeichert, sodass angemeldeten Benutzern speziell auf sie zugeschnittene Seiten angezeigt werden können.

17

WEITERE FUNKTIONEN HINZUFÜGEN

Im letzten Kapitel erfahren Sie, wie Sie suchmaschinenfreundliche URLs anlegen können. Sie lernen zudem, wie Sie Ihren Mitgliedern die Möglichkeit geben, Kommentare zu Artikeln hinzuzufügen und zu signalisieren, dass Ihnen diese gefallen - beides sind gängige Funktionen in sozialen Netzwerken.

14

REFACTORING & DEPENDENCY INJECTION

Unter Refactoring versteht man die Verbesserung des Codes einer Anwendung durch Umstrukturierung, ohne die Funktionen oder das Verhalten der Anwendung zu ändern.

Websites bestehen oft aus Tausenden von Codezeilen, sodass deren Organisation sehr wichtig ist.

Wenn sich Websites vergrößern und neue Funktionen hinzugefügt werden, ist es hilfreich, den Code zu überarbeiten und zu refaktorieren.

Dabei wird die Struktur des Codes so verbessert, dass er einfacher

- zu lesen,
- zu verwalten und
- um neue Funktionen zu erweitern ist.

In diesem Kapitel ändert sich die Funktionalität der Website, die Sie im vorigen Kapitel kennengelernt haben, nicht, und die Benutzeroberfläche bleibt unverändert, aber der SQL- und der PHP-Code für die Arbeit mit der Datenbank werden aus den vom Benutzer angeforderten PHP-Seiten in eine Reihe von Klassen verschoben.

Die PHP-Seiten erstellen Objekte unter Verwendung dieser Klassen und nutzen deren Methoden, um in der Datenbank gespeicherte Daten abzurufen oder zu ändern. Die Klassen fassen den Code, der für die Arbeit mit der Datenbank benötigt wird, zusammen (anstatt ihn über die gesamte Website zu verteilen), was die Wartung des Codes und das Hinzufügen neuer Funktionen erleichtert. In diesem Kapitel werden auch zwei neue Programmietechniken vorgestellt:

- Mit **Dependency Injection (DI)** wird sichergestellt, dass Code, der auf die Datenbank zugreifen muss, stets ein PDO-Objekt enthält, das für die Arbeit mit der Datenbank verwendet werden kann.
- **Autoloading** weist den PHP-Interpreter an, Dateien mit Klassendefinitionen nur dann zu laden, wenn eine Klasse zur Erstellung eines Objekts benötigt wird, anstatt `include-` oder `require-`Anweisungen zu verwenden, die die Klassendateien jedes Mal laden, wenn die Seite angefordert wird (auch wenn sie am Ende nicht zur Erstellung eines Objekts verwendet werden).

Am Ende des Kapitels ist der Code für das CMS auf mehr Dateien aufgeteilt als in der vorherigen Version, aber der Code für die einzelnen Aufgaben ist leichter zu finden.



A PAPETERIE POUR FEMME CHARMANTE[®]
La collection apprécier de tous

ordonné

Carnet in-folio, format A5
quadrillé 7mm/32 feuillets

L. 148mm x H. 210mm

80 g
Papier
Satiné
PETIT GRIS PUR

170 g
BLACKWING • PEARL

WINSON • NEWTON
ENGLAND

OBJEKTE FÜR DIE ARBEIT MIT DER DATENBANK VERWENDEN

Die Beispielseite verwendet drei Grundkonzepte: Artikel, Kategorien und Mitglieder. Für jedes wird eine Klasse verwendet.

In Kapitel 13 enthielt jede PHP-Seite der Beispielseite die benötigten SQL-Anweisungen, um Daten aus der Datenbank abzurufen oder zu ändern. Diese SQL-Anweisungen wurden dann durch den Aufruf der benutzerdefinierten, in Kapitel 12 vorgestellten Funktion `pdo()` ausgeführt.

In diesem Kapitel wurden die SQL-Anweisungen und der Code, mit dem sie aufgerufen werden, in drei Klassen ausgelagert:

- `Article` verfügt über Methoden, die Artikeldaten in der Datenbank abrufen, erstellen, aktualisieren oder löschen.
- `Category` verfügt über Methoden, die Kategorie-daten in der Datenbank abrufen, erstellen, aktualisieren oder löschen.
- `Member` verfügt über Methoden, die Mitglieds-daten abrufen.

Jede Klassendefinition befindet sich in einer eigenen Datei im Ordner `src/classes`. Die Dateinamen sind die Klassennamen, gefolgt von der Dateierweiterung `.php` (z.B. befindet sich die Klasse `Article` in der Datei `article.php`).

HINWEIS: Die Namenskonventionen in diesem Kapitel folgen den Regeln, die von der Gruppe PHP Framework Interoperability Group (PHP-FIG) festgelegt wurden. Diese Gruppe setzt sich aus erfahrenen PHP-Entwicklern zusammen, die etablierte PHP-Projekte erstellen und pflegen. Ihr Ziel ist es, die Zusammenarbeit zwischen diesen Projekten zu fördern, aber ihre Richtlinien werden auch von vielen anderen PHP-Entwicklern übernommen. Weitere Informationen finden Sie unter <http://php-fig.org>.

In den vorangegangenen Kapiteln enthielt jede Seite die SQL-Abfragen für die benötigten Daten. Deshalb wurden oft auf mehreren Seiten sehr ähnliche Abfragen verwendet.

Beispielsweise wurden für die Seite, über die die Webnutzer einen Artikel betrachten konnten, und die Seite, über die Administratoren einen Artikel bearbeiten konnten, sehr ähnliche SQL-Abfragen verwendet.

In diesem Kapitel verwenden beide Seiten dieselbe `get()`-Methode der Klasse `Article`.

Article

EIGENSCHAFT	BESCHREIBUNG
<code>\$db</code>	Database-Objekt speichern

METHODE	BESCHREIBUNG
<code>get()</code>	Artikel abrufen
<code>getAll()</code>	Artikelzusammenfassungen abrufen
<code>count()</code>	Artikelgesamtzahl zurückgeben
<code>create()</code>	neuen Artikel erstellen
<code>update()</code>	bestehenden Artikel aktualisieren
<code>delete()</code>	Artikel löschen
<code>imageDelete()</code>	Bild aus dem Artikel löschen
<code>altUpdate()</code>	Alt-Text für Bild aktualisieren
<code>search()</code>	Artikel suchen
<code>searchCount()</code>	Anzahl der Suchtreffer

Jede Klassendefinition besitzt ihre eigene PHP-Datei im Ordner `src/classes`. Der Dateiname ist der Name der Klasse, gefolgt von der Dateierweiterung `.php`.

Wie Sie sehen werden, verfügen einige Methoden über Parameter, die z. B. festlegen, ob alle Artikel zurückgegeben werden oder nur diejenigen, die

- veröffentlicht worden sind,
- zu einer bestimmten Kategorie gehören,
- von einem bestimmten Autor erstellt wurden.

Andere Parameter steuern beispielsweise, wie viele Zeilen der Ergebnismenge hinzugefügt werden sollen.

Diese drei Klassen werden nur verwendet, um Objekte bei Bedarf zu erstellen. Zum Beispiel erstellt jede Seite ein `Category`-Objekt für die Kategorien, die in der Navigation angezeigt werden sollen. Ein `Member`-Objekt wird hingegen nur erstellt, wenn eine Seite das Profil eines Mitglieds anzeigt.

Diese Klassen haben alle die Eigenschaft `$db`. Wenn ein Objekt mit ihnen erstellt wird, speichert die Eigenschaft `$db` ein Objekt, mit dem eine Verbindung zur Datenbank hergestellt und die SQL-Anweisungen ausgeführt werden können.

Category

EIGENSCHAFT	BESCHREIBUNG
<code>\$db</code>	Database-Objekt speichern
METHODE	BESCHREIBUNG
<code>get()</code>	Kategorie abrufen
<code>getAll()</code>	alle Kategorien abrufen
<code>count()</code>	Gesamtzahl der Kategorien zurückgeben
<code>create()</code>	neue Kategorie erstellen
<code>update()</code>	bestehende Kategorie aktualisieren
<code>delete()</code>	Kategorie löschen

Member

EIGENSCHAFT	BESCHREIBUNG
<code>\$db</code>	Database-Objekt speichern
METHODE	BESCHREIBUNG
<code>get()</code>	Mitglied abrufen
<code>getAll()</code>	alle Mitglieder abrufen

HINWEIS: Falls noch nicht geschehen, öffnen Sie die Datei `config.php` im Ordner `resources` und aktualisieren Sie die Werte der Variablen, die für die Verbindung zur Datenbank verwendet werden. Sie sollten dieselben Angaben verwenden, die in den Kapiteln 12 und 13 verwendet wurden.

Wenn Sie sich die Website ansehen, sieht sie genau so aus wie die vorherige Version in Kapitel 13, aber sie nutzt die neuen Klassen, die Sie in diesem Kapitel kennengelernt.

DATENBANK-OBJEKT

Das Database-Objekt wird mithilfe der benutzerdefinierten Klasse `Database` erstellt. Die PDO-Klasse wird dadurch erweitert, und es bieten sich dieselben Eigenschaften und Methoden sowie eine neue `runSQL()`-Methode, die zur Ausführung von SQL-Anweisungen verwendet wird.

Im vorherigen Abschnitt enthielt jede PHP-Seite die Datei `database-connection.php`, die ein in der Variablen `$pdo` gespeichertes PDO-Objekt erstellte. Anschließend riefen die Seiten die benutzerdefinierte Funktion `pdo()` auf (definiert in `functions.php`), um SQL-Anweisungen mit diesem PDO-Objekt auszuführen.

In diesem Kapitel wird mit einer neuen Klasse `Database` ein Database-Objekt erstellt. Diese Klasse erweitert das PDO-Objekt, was bedeutet, dass das Database-Objekt alle Methoden und Eigenschaften des PDO-Objekts erbt. Sie fügt dann die weitere Methode `runSQL()` hinzu, die dieselbe Aufgabe erfüllt wie die in den Kapiteln 12 und 13 verwendete benutzerdefinierte Funktion `pdo()`.

Das neue Database-Objekt ist also nur ein PDO-Objekt mit der zusätzlichen Methode `runSQL()` (die der Code zur Ausführung von SQL-Anweisungen verwendet).

Die auf den beiden vorherigen Seiten gezeigten Objekte `Article`, `Category` und `Member` benötigen ein PDO-Objekt (oder sind von diesem abhängig), um eine Verbindung zur Datenbank herzustellen. Daher spricht man beim PDO-Objekt von einer **Abhängigkeit**.

Wie Sie auf S. 530–531 gesehen haben, speichert der PHP-Interpreter, wenn er ein Objekt erstellt und es in einer Variablen oder einer Objekteigenschaft speichert, eigentlich den Speicherort des Objekts im Speicher des PHP-Interpreters (nicht das Objekt selbst). Daher können mehrere Variablen oder Eigenschaften den Ort eines einzigen Objekts speichern.

Wenn die Seite ein `Article`-, `Category`- oder `Member`-Objekt erstellt, wird der Ort des Database-Objekts in der Eigenschaft `$db` dieses Objekts gespeichert.

Wenn die Seite dann ein anderes dieser Objekte (`Article`, `Category` oder `Member`) erstellt, speichert dieses Objekt ebenfalls den Ort desselben Database-Objekts in seiner `$db`-Eigenschaft. Das bedeutet, dass die Objekte `Article`, `Category` und `Member` alle dasselbe Database-Objekt nutzen können (und wie Sie gesehen haben, ist das Database-Objekt ein PDO-Objekt mit einer zusätzlichen Methode).

Man sagt, dass die Abhängigkeit in das `Article`-, `Category`- und `Member`-Objekt injiziert wurde, sodass jedes von Ihnen sie nutzen kann. Aus diesem Grund wird diese Technik als Dependency Injection bezeichnet. Die gemeinsame Nutzung einer Datenbankverbindung von allen Objekten einer Seite ist hilfreich, da der PHP-Interpreter jeweils nur eine begrenzte Anzahl von Verbindungen zur Datenbank herstellen kann.

Database

METHODE	BESCHREIBUNG
<code>runSQL()</code>	führt eine SQL-Anweisung aus

HINWEIS: Vererbung und Dependency Injection sind Beispiele für Entwurfsmuster; Lösungen, die auf allgemeine Programmierprobleme angewendet werden können. Wenn Sie mehr Klassen verwenden, ist es hilfreich, weitere Entwurfsmuster zu lernen. Manche bevorzugen zum Beispiel das Muster Komposition.

CONTAINER-OBJEKT

Die Objekte Article, Category, Member und Database werden mithilfe von Methoden eines fünften Objekts erstellt, das als Container-Objekt bezeichnet wird. Es heißt so, weil seine Eigenschaften die anderen Objekte enthalten.

In der Einführung zu diesem Abschnitt haben Sie gesehen, dass jede Seite die Datei `bootstrap.php` enthält (S. 529). Diese Datei erstellt ein CMS-Objekt mit einer Klasse CMS und speichert es in der Variablen `$cms`.

Wenn das CMS-Objekt erstellt wird, wird seine `__construct()`-Methode ausgeführt. Diese erstellt ein Database-Objekt und speichert es in der `$db`-Eigenschaft des CMS-Objekts (weil jede Seite der Site auf die Datenbank zurückgreift).

Wenn eine Seite auf Artikel-, Kategorie- oder Mitgliederdaten zugreifen muss, erstellt sie unter Verwendung der Methoden `getArticle()`, `getCategory()` und `getMember()` des CMS-Objekts ein Article-, Category- oder Member-Objekt. Diese Objekte nutzen alle dasselbe Database-Objekt.

CMS

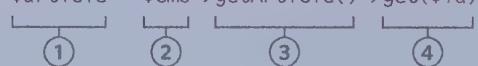
EIGENSCHAFT	BESCHREIBUNG
<code>\$db</code>	speichert ein Database-Objekt
<code>\$article</code>	speichert ein Article-Objekt
<code>\$category</code>	speichert ein Category-Objekt
<code>\$member</code>	speichert ein Member-Objekt

METHODE	BESCHREIBUNG
<code>getArticle()</code>	Gibt ein Article-Objekt zurück
<code>getCategory()</code>	Gibt ein Category-Objekt zurück
<code>getMember()</code>	Gibt ein Member-Objekt zurück

Eines der Ziele der Verwendung dieser Klassen ist es, sicherzustellen, dass die gesamte Arbeit mit der Datenbank über das CMS-Objekt und die darin enthaltenen Objekte erfolgt (und nicht über die vom Benutzer angeforderten PHP-Seiten verteilt wird).

Sobald `bootstrap.php` ein CMS-Objekt erstellt hat, kann jede Seite mit nur einer Anweisung Daten in der Datenbank abrufen oder ändern. Die folgende Anweisung ruft Daten über einen Artikel ab und speichert sie in der Variablen `$article`.

```
$article = $cms->getArticle()->get($id);
```



Dabei wird die Methodenverkettung (S. 457) verwendet: Wenn eine Methode ein Objekt zurückgibt (wie die Methode `getArticle()` des CMS-Objekts), können Sie in derselben Anweisung eine Methode des zurückgegebenen Objekts aufrufen.

1. Die Variable `$article` enthält die Artikeldaten, die von der Datenbank zurückgegeben werden.
2. Die Datei `bootstrap.php` hat bereits ein CMS-Objekt erstellt und in der Variablen `$cms` gespeichert.
3. Die Methode `getArticle()` des CMS-Objekts gibt ein Article-Objekt zurück.
4. Die `get()`-Methode des Article-Objekts gibt die Daten über einen einzelnen Artikel als Array zurück (das in der Variablen `$article` gespeichert ist).

CMS-CONTAINER-OBJEKT

Jede Seite enthält die Datei `bootstrap.php`, die ein CMS-Objekt erstellt. Die Methoden des CMS-Objekts erstellen die Objekte `Article`, `Category` und `Member` und ermöglichen ihnen die Nutzung desselben Database-Objekts.

1. Die CMS-Klasse beginnt mit vier Eigenschaften, die den Ort der vier anderen Objekte speichern, die das CMS-Objekt enthalten kann. Jede dieser Eigenschaften wird mit dem Schlüsselwort `protected` deklariert, sodass die gespeicherten Werte nur von Code in diesem Objekt abgerufen werden können. Alle Eigenschaften haben den Standardwert `null`.

2. Wenn ein Objekt mit der CMS-Klasse erstellt wird, wird automatisch seine `__construct()`-Methode ausgeführt. Diese benötigt drei Informationen, damit sie ein Database-Objekt für die Datenbankverbindung erstellen kann (sie wurden in der Datei `config.php` auf S. 528 gespeichert):

- Der DSN speichert den Speicherort der Datenbank.
- Der Benutzername und das Passwort eines Datenbank-Benutzerkontos werden zur Anmeldung bei der Datenbank verwendet.

3. Die `__construct()`-Methode erstellt ein neues Objekt mit der Database-Klasse. Wie Sie gerade gesehen haben, ist ein Database-Objekt ein PDO-Objekt, das um eine zusätzliche Methode erweitert wurde.

Die Argumente, die zum Erstellen eines Database-Objekts benötigt werden, sind dieselben wie die zum Erstellen eines PDO-Objekts; es werden der DSN, ein Benutzername und ein Passwort benötigt, um sich mit der Datenbank zu verbinden (die bei der Erstellung des Objekts in `bootstrap.php` angegeben wurden, siehe S. 529).

Das erstellte Database-Objekt wird in der Eigenschaft `$db` dieses CMS-Objekts gespeichert.

Anschließend werden mit den drei Methoden die `Article`-, `Category`- und `Member`-Objekte zurückgegeben, die das CMS-Objekt enthalten kann. Beachten Sie, dass jede Methode mit einer `if`-Anweisung beginnt; dadurch wird sichergestellt, dass eine Seite nie mehr als eines dieser Objekte erzeugt.

4. Die Methode `getArticle()` ist definiert. Sie gibt ein `Article`-Objekt zurück, das zum Abrufen oder Ändern von Artikeldaten verwendet wird.

5. Eine `if`-Anweisung prüft, ob die Eigenschaft `$article` dieses CMS-Objekts den Wert `null` hat. In diesem Fall hat diese Seite noch kein `Article`-Objekt erstellt; es muss erstellt werden, bevor es von dieser Methode zurückgegeben werden kann.

6. Ein `Article`-Objekt wird mithilfe der `Article`-Klasse erstellt. Der Speicherort des Database-Objekts wird an die Konstruktormethode der Klasse `Article` übergeben (sodass das von ihr erstellte `Article`-Objekt das in der Eigenschaft `$db` gespeicherte Database-Objekt für den Zugriff auf die Datenbank verwenden kann). Das erstellte `Article`-Objekt wird dann in der Eigenschaft `$article` dieses Objekts gespeichert.

7. Das `Article`-Objekt in der `$article`-Eigenschaft dieses Objekts wird zurückgegeben, damit es von dem Code verwendet werden kann, der die `getArticle()`-Methode aufgerufen hat.

Die Methoden `getCategory()` und `getMember()` funktionieren genauso wie die Methode `getArticle()`, aber sie geben ein `Category`- oder `Member`-Objekt zurück, um mit `Category`- oder `Member`-Daten zu arbeiten.

```

<?php
class CMS
{
    protected $db      = null;           // Speichert Verweis auf Datenbankobjekt
    protected $article = null;           // Speichert Verweis auf Artikelobjekt
    protected $category = null;          // Speichert Verweis auf Kategorieobjekt
    protected $member   = null;           // Speichert Verweis auf Mitgliederobjekt

    public function __construct($dsn, $username, $password)
    {
        $this->db = new Database($dsn, $username, $password); // Datenbankobjekt erst.
    }

    public function getArticle()
    {
        if ($this->article === null) {
            $this->article = new Article($this->db); // Artikelobjekt erstellen
        }
        return $this->article;                      // Rückgabe Artikelobjekt
    }

    public function getCategory()
    {
        if ($this->category === null) {             // Falls $category-Eigenschaft null
            $this->category = new Category($this->db); // Kategorie-Objekt erzeugen
        }
        return $this->category;                     // Rückgabe Category-Objekt
    }

    public function getMember()
    {
        if ($this->member === null) {              // Falls $member-Eigenschaft null
            $this->member = new Member($this->db); // Member-Objekt erzeugen
        }
        return $this->member;                      // Rückgabe Member-Objekt
    }
}

```

Dieses grundlegende Container-Objekt dient dazu, die beteiligten Konzepte vorzustellen. Ein Database-Objekt wird im Konstruktor der CMS-Klasse erstellt, da jede Seite der Site eine Verbindung zur Datenbank herstellt. Das Database-Objekt könnte auch bei Bedarf mit einer `getDatabase()`-Methode erstellt werden (wie bei den Objekten Article, Category und Member), aber die Verbindungsdaten müssten dann in Eigenschaften des CMS-Objekts gespeichert werden.

Der DSN, der Benutzername und das Passwort werden mit drei Parametern an den Konstruktor der CMS-Klasse übergeben, da es in diesem Buch um die Verwendung von PHP mit MySQL geht und die Verbindungsdaten sich nicht ändern werden. Einige Programmierer speichern diese Informationen in einem separaten Konfigurationsobjekt, sodass die Site problemlos mit einer anderen Art von Datenbank arbeiten kann. Diese Entscheidungen hängen vom Umfang des Projekts ab.

DATABASE-KLASSE

Die Klasse Database erweitert das PDO-Objekt und fügt die zusätzliche Methode runSQL() hinzu, die zur Ausführung von SQL-Anweisungen verwendet wird (genau wie die Funktion pdo(), die in den Kapiteln 12 und 13 verwendet wurde).

Die Klasse Database erweitert die in PHP integrierte PDO-Klasse, was bedeutet, dass sie alle Eigenschaften, Methoden und Konstanten der PDO-Klasse erbt, die mit den Schlüsselwörtern public oder protected deklariert wurden. Die Klasse Database fügt die zusätzliche Methode runSQL() zu diesem Objekt hinzu, was verwendet wird, um SQL-Anweisungen auszuführen. Hierbei

- wird die PDO-Klasse als übergeordnete Klasse bezeichnet
- ist die Database-Klasse die untergeordnete Klasse
Bei der Erweiterung eines Objekts wird häufig als gute Praxis angesehen, sicherzustellen, dass die untergeordnete Klasse die übergeordnete Klasse überall dort ersetzen kann, wo sie verwendet wird, und dass der Code genau gleich ausgeführt wird (da er der übergeordneten Klasse Funktionen hinzufügt).

1. Auf den Klassennamen (Database) folgt das Schlüsselwort extends, dann der Name der Klasse, die erweitert wird (in diesem Fall die integrierte PDO-Klasse).

2. Die Methode __construct() wird automatisch ausgeführt, wenn ein Objekt mit dieser Klasse erstellt wird. Sie hat vier Parameter:

- den DSN, der den Speicherort der Datenbank enthält
- den Benutzernamen eines Datenbank-Benutzerkontos
- das Passwort für das Datenbank-Benutzerkonto
- ein optionales Array, das PDO-Einstellungen enthalten kann

Wenn das Database-Objekt in der __construct()-Methode des CMS-Objekts erstellt wird (siehe vorherige Seite), werden Argumente für ...

- die ersten drei Parameter angegeben, weil sie sich jedes Mal ändern, wenn der Code zur Ausführung einer Website verwendet wird
- den vierten Parameter nicht angegeben, da der Benutzer diese Optionen in der Beispielseite nicht einstellen soll

Der vierte Parameter wird jedoch der Konstruktorfunktion der Klasse Database hinzugefügt (auch wenn er nicht verwendet wird), da sie die gleichen Parameter wie die PDO-Klasse haben sollte. Dadurch (und durch die nächsten beiden Schritte) wird sichergestellt, dass die untergeordnete Klasse Database überall dort verwendet werden kann, wo die übergeordnete PDO-Klasse verwendet wird.

3. Ein Array mit dem Namen \$default_options enthält Optionen, die zum Erstellen des PDO-Objekts verwendet werden und ihm die folgenden Anweisungen geben:

- den Standardabrufmodus auf ein assoziatives Array setzen
- den Modus emulate准备es ausschalten (um sicherzustellen, dass die Daten mit dem richtigen Datentyp zurückgegeben werden)
- eine Ausnahme erstellen, wenn ein Problem auftritt

4. Mit der PHP-Funktion array_replace() werden dann alle Werte im Array \$default_options durch Werte ersetzt, die für den Parameter \$options angegeben wurden. Auf der Beispielseite werden keine Werte für dieses Array angegeben, sodass immer die Standardoptionen verwendet werden (dies bedeutet jedoch, dass die untergeordnete Klasse Database überall dort verwendet werden kann, wo die übergeordnete PDO-Klasse verwendet wird).

```

<?php
① class Database extends PDO
{
    public function __construct(string $dsn, string $username, string $password,
                                array $options = [])
    {
        // Set default PDO options
        $default_options[PDO::ATTR_DEFAULT_FETCH_MODE] = PDO::FETCH_ASSOC;
        $default_options[PDO::ATTR_EMULATE_PREPARES] = false;
        $default_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
        $options = array_replace($default_options, $options); // Defaults ersetzen
        parent::__construct($dsn, $username, $password, $options); // PDO-Objekt erst.
    }

    ⑥ public function runSQL(string $sql, $arguments = null)
    {
        if (!$arguments) { // Wenn keine Argumente
            ⑦ return $this->query($sql); // SQL ausführen, PDOStatement-Objekt zurückgeben
        }
        ⑧ $statement = $this->prepare($sql); // Falls noch ausgeführt
        ⑨ $statement->execute($arguments); // Anweisung mit Argumenten ausführen
        ⑩ return $statement; // PDOStatement-Objekt zurückgeben
    }
}

```

5. Da das Database-Objekt das PDO-Objekt erweitert, wird die Methode `__construct()` der Database-Klasse ausgeführt, wenn ein Objekt mit dieser Klasse erstellt wird. Die Methode `__construct()` der PDO-Klasse wird jedoch nicht automatisch ausgeführt (und es ist die Methode `__construct()` der PDO-Klasse, die die Verbindung zur Datenbank herstellt).

Daher muss die Klasse `Database` der `__construct()`-Methode der übergeordneten PDO-Klasse mitteilen, dass sie ausgeführt werden soll, sodass eine Verbindung zur Datenbank hergestellt werden kann.

Sie erhält die gleichen Argumente, die das PDO-Objekt benötigt, um eine Verbindung zur Datenbank herzustellen.

6. Die `runSQL()`-Methode ist fast identisch mit der in den Kapiteln 12 und 13 verwendeten `pdo()`-Funktion. Der einzige Unterschied ist, dass sie kein PDO-Objekt als Argument benötigt, da dieses Objekt die PDO-Klasse erweitert.

`runSQL()` hat zwei Parameter: die auszuführende SQL-Anweisung und ein Array von Daten, die als Ersatz für die Platzhalter in der SQL-Anweisung verwendet werden sollen.

7. Wenn für die SQL-Anweisung keine Argumente angegeben wurden, wird die Methode `query()` von PDO mit der SQL-Anweisung als Argument aufgerufen. Die `query()`-Methode führt die SQL-Anweisung aus, und ein PDOStatement-Objekt, das die Ergebnismenge repräsentiert, wird zurückgegeben.

8. Wenn Argumente angegeben wurden, wird die PDO-Methode `prepare()` aufgerufen. Sie gibt ein PDOStatement-Objekt zurück, das die SQL-Anweisung repräsentiert.

9. Die `execute()`-Methode des PDOStatement-Objekts wird aufgerufen, um die Anweisung auszuführen.

10. Ein PDOStatement-Objekt, das die Ergebnismenge repräsentiert, wird von der Methode zurückgegeben.

CATEGORY-KLASSE

Die Klasse Category fasst die SQL-Anweisungen und den PHP-Code zusammen, die zum Abrufen und Ändern von Kategoriedaten in der Datenbank erforderlich sind. Sie speichert einen Verweis auf das Database-Objekt in ihrer Eigenschaft \$db.

In den Kapiteln 12 und 13 befanden sich die SQL-Anweisungen und der Code zum Abrufen oder Ändern von Datenbankdaten in den einzelnen aufgerufenen PHP-Seiten. Das Verschieben desselben Codes in Methoden einer Klasse hat drei Vorteile:

A. Jede Seite, die Daten in der Datenbank abrufen oder ändern muss, kann dies in einer einzigen Anweisung tun. Zum Beispiel ruft die folgende Anweisung die Details einer Kategorie ab und speichert sie in einer Variablen:

```
$category = $CMS->category()->get($id);
```

B. Sie verhindert die Wiederholung von ähnlichem Code in mehreren Dateien. Zum Beispiel benötigen mehrere Dateien Details zu allen Kategorien aus der Datenbank:

- Jede öffentliche Seite sammelt die Kategorien, die in der Hauptnavigation der Website erscheinen.
- Die Verwaltungsseite categories.php listet alle Kategorien auf, die vom Admin aktualisiert oder gelöscht werden können.
- Die Verwaltungsseite article.php zeigt alle Kategorien in einer Drop-down-Auswahlbox an, sodass der Admin auswählen kann, zu welcher Kategorie ein Artikel gehört.

Alle diese Seiten können die eine getAll()-Methode dieser Klasse verwenden, um Daten über alle Kategorien zu erhalten.

C. Wenn die SQL-Anweisungen zum Abrufen oder Ändern von Kategoriedaten aktualisiert werden müssen, kann diese eine Datei geändert werden (anstatt jede Datei zu ändern, die Kategoriedaten abruft), was die Wartung des Codes erleichtert.

Wie Sie auf S. 540–542 gesehen haben, wird ein Category-Objekt beim ersten Aufruf der getCategory()-Methode des CMS-Objekts erstellt. Die rechte Seite zeigt die Methoden, die Kategoriedaten aus der Datenbank abrufen (Methoden zum Ändern von Kategoriedaten werden als Nächstes gezeigt).

1. Die Eigenschaft \$db wird erstellt, um den Ort des Database-Objekts zu speichern. Sie wird als geschützte Eigenschaft deklariert, damit sie nur von dem Code innerhalb dieser Klasse verwendet wird.
2. Die Methode __construct() wird ausgeführt, wenn ein Category-Objekt mit dieser Klasse erstellt wird. Ihr einziger Parameter ist ein Verweis auf das Database-Objekt (die Typdeklaration stellt sicher, dass das Objekt mit der Database-Klasse erstellt wurde).
3. Der Speicherort des Database-Objekts wird in der Eigenschaft \$db des Category-Objekts gespeichert, sodass die anderen Methoden dieses Objekts es verwenden können.
4. Die Methode get() enthält den Code, um Daten über eine Kategorie aus der Datenbank abzurufen. Sie hat einen Parameter: die ID der Kategorie, die abgerufen werden soll.

(Der Code in dieser Methode ist fast identisch mit dem Code, der zum Abrufen einer Kategorie in den Kapiteln 12 und 13 verwendet wird.)

5. Die SQL-Abfrage, die zum Abrufen der Daten aus der Datenbank benötigt wird, wird in der Variablen \$sql gespeichert. Dies ist dieselbe SQL-Abfrage, die sowohl in der Datei category.php im öffentlichen Teil der Website als auch in der Datei category.php auf den Verwaltungsseiten verwendet wurde.

```

<?php
class Category
{
    ①   protected $db;                                // Verweis auf Datenbankobjekt

    ②   public function __construct(Database $db)
    {
        ③       $this->db = $db;                          // Speichert Verweis auf Datenbankobjekt
    }

    ④   public function get(int $id)
    {
        ⑤       $sql = "SELECT id, name, description, navigation
                    FROM category
                    WHERE id = :id;";                      // SQL zum Abrufen einer Kategorie
        ⑥       return $this->db->runSQL($sql, [$id])->fetch(); // Kategoriedaten zurück
    }

    ⑦   public function getAll(): array
    {
        ⑧       $sql = "SELECT id, name, navigation
                    FROM category;";                     // Alle Kategorien
        ⑨       return $this->db->runSQL($sql)->fetchAll(); // Alle Kategorien zurück
    }

    ⑩  public function count(): int
    {
        ⑪     $sql = "SELECT COUNT(id) FROM category;"; // Kategorien zählen
        ⑫     return $this->db->runSQL($sql)->fetchColumn(); // Kategorieanzahl zurück
    }
}

```

6. Um die SQL-Anweisung auszuführen, wird über die Eigenschaft `$db` dieses Objekts mit `$this->db` auf das Database-Objekt zugegriffen. Die Methode `runSQL()` des Database-Objekts wird aufgerufen, um die SQL-Anweisung auszuführen. Sie funktioniert wie die Funktion `pdo()`, die in den Kapiteln 12 und 13 verwendet wurde, und hat zwei Parameter:

- die SQL-Anweisung, die ausgeführt werden soll
- Daten zum Ersetzen von Platzhaltern in der SQL-Anweisung

Die Methode `runSQL()` gibt ein PDOStatement-Objekt zurück, das die erzeugte Ergebnismenge repräsentiert. Die `fetch()`-Methode des PDOStatement-Objekts zieht dann die Kategoriedaten aus der Ergebnismenge als Array, und dieses Array wird von der Methode zurückgegeben.

7. `getAll()` liefert Details zu allen Kategorien.

8. `$sql` enthält den SQL-Code zum Abrufen aller Kategoriedaten.

9. Die SQL-Anweisung wird ausgeführt, und die Methode `fetchAll()` des PDOStatement-Objekts ruft alle Kategoriedaten aus der Ergebnismenge als Array ab. Dieses Array wird dann von der Methode zurückgegeben.

10. `count()` gibt die Anzahl der Kategorien zurück.

11. `$sql` enthält den SQL-Code, um die Gesamtzahl der Kategorien zu ermitteln.

12. Die SQL-Anweisung wird ausgeführt, und `fetchColumn()` liefert die Anzahl der Kategorien aus der Ergebnismenge.

KATEGORIEN ERSTELLEN, AKTUALISIEREN UND LÖSCHEN

Drei Methoden erstellen, aktualisieren und löschen Kategorien. Mit try...catch-Blöcken werden Situationen behandelt, in denen ein Benutzer einen doppelten Kategorienamen angibt oder versucht, eine Kategorie zu löschen, die Artikel enthält.

Auf der rechten Seite sehen Sie die Methoden zum Erstellen, Aktualisieren und Löschen von Kategorien.

1. Mit der Methode `create()` wird eine neue Kategorie erstellt.

Sie hat als Parameter ein Array, das für jeden Parameter der SQL-Anweisung (Name der Kategorie, Beschreibung und ob sie in der Navigation angezeigt werden soll) ein Element enthalten muss. Sie gibt zurück:

- `true`, wenn die Kategorie erstellt wurde
- `false`, wenn der Titel der Kategorie bereits verwendet wurde

Ansonsten wird eine Ausnahme ausgelöst. Der Code ist dem in `admin/category.php` in Kapitel 13 verwendeten Code sehr ähnlich.

2. Der Code zum Erstellen der Kategorie befindet sich in einem `try`-Block.

3. `$sql` speichert den SQL-Code zum Erstellen einer Kategorie.

4. Die SQL-Anweisung wird mit der `runSQL()`-Methode des `Database`-Objekts ausgeführt.

5. Wenn keine Ausnahme ausgelöst wurde, hat der Code funktioniert, und die Funktion gibt `true` zurück, um den Erfolg anzuzeigen.

6. Wenn eine Ausnahme ausgelöst wurde, wird der `catch`-Block ausgeführt, um die Ausnahme zu behandeln.

7. Wenn der Fehlercode des Ausnahmeobjekts 1062 lautet, weist dies auf einen doppelten Eintrag hin, da der Kategorietitel bereits verwendet wird, und die Funktion gibt `false` zurück.

8. Wenn es sich um einen anderen Fehlercode handelt, wird die Ausnahme erneut ausgelöst. Sie wird dann

von der Standardfunktion zur Behandlung von Ausnahmen behandelt.

9. Die `update()`-Methode aktualisiert eine bestehende Kategorie. Sie funktioniert genauso wie die `create()`-Methode.

Die einzigen Unterschiede sind folgende:

- Das Array, das der Methode übergeben wird, muss die ID der Kategorie sowie ihren Namen, ihre Beschreibung und die Angabe, ob sie in der Navigation enthalten sein soll oder nicht, enthalten.
- Die SQL-Anweisung aktualisiert die Kategorie und verwendet eine `UPDATE`-Anweisung, keine `CREATE`-Anweisung.

10. Die `delete()`-Methode ist ähnlich, aber:

- Sie benötigt nur die ID der zu löschen Kategorie.
- Die SQL-Anweisung verwendet einen `DELETE`-Befehl.
- Der `catch`-Block prüft, ob der Fehlercode 1451 lautet; dieser Fehlercode weist auf eine Integritätsbedingung hin: In der Kategorie gibt es Artikel, die in eine andere Kategorie verschoben oder gelöscht werden müssen, bevor diese Kategorie gelöscht werden kann.

Diese Methoden funktionieren wie der Code, der in den einzelnen PHP-Seiten der Website in Kapitel 13 enthalten war.

Durch das Verschieben des Codes in eine Klasse wird jedoch die Codemenge in den PHP-Seiten reduziert, die die Kategoriedaten abrufen oder ändern. Es werden mehrere Dateien eingespart, die ähnlichen Code wiederholen. Wenn der Website neue Funktionen hinzugefügt werden (weiter hinten im Buch), werden neue Methoden zu den Klassen hinzugefügt, um neue Aufgaben zu implementieren.

```
① public function create(array $category): bool
{
    ②     try {                                // Versuch, Kategorie zu erstellen
        ③         $sql = "INSERT INTO category (name, description, navigation)
                    VALUES (:name, :description, :navigation);"; // Kategorie hinzuf.
        ④         $this->db->runSQL($sql, $category); // Neue Kategorie
        ⑤         return true;                      // Hat funktioniert, true
    } catch (PDOException $e) {                // Wenn Ausnahme ausgelöst
        ⑥         if ($e->errorInfo[1] == 1062) {      // Wenn Fehler doppelter Eintrag
            ⑦             return false;           // false zurückgeben
        } else {
            ⑧             throw $e;               // Wenn andere Ausnahme
                                            // Ausnahme erneut auslösen
        }
    }
}

⑨ public function update(array $category): bool
{
    try {                                    // Versuch, Kategorie zu aktualisieren
        $sql = "UPDATE category
                    SET name = :name, description = :description, navigation = :navigation
                    WHERE id = :id;";           // Kategorie aktualisieren
        $this->db->runSQL($sql, $category);
        return true;                          // Hat funktioniert, true
    } catch (PDOException $e) {              // Wenn Ausnahme ausgelöst
        if ($e->errorInfo[1] == 1062) {      // Wenn doppelter Eintrag
            return false;                   // false zurückgeben
        } else {
            throw $e;                     // Wenn andere Ausnahme
                                            // Ausnahme erneut auslösen
        }
    }
}

⑩ public function delete(int $id): bool
{
    try {                                    // Versuch, Kategorie zu löschen
        $sql = "DELETE FROM category WHERE id = :id;"; // Kategorie löschen
        $this->db->runSQL($sql, [$id]);
        return true;                          // Hat funktioniert, true
    } catch (PDOException $e) {              // Wenn Ausnahme ausgelöst
        if ($e->errorInfo[1] == 1451) {      // Wenn Integritätsbedingung
            return false;                   // false zurückgeben
        } else {
            throw $e;                     // Wenn andere Ausnahme
                                            // Ausnahme erneut auslösen
        }
    }
}
```

ARTIKELDATEN ABRUFEN

Die Methoden zum Abrufen von Artikeldaten in der Klasse Article verwenden Parameter, mit deren Hilfe eine Methode unterschiedliche Artikeldaten für verschiedene Seiten abrufen kann.

1. Die Methode `get()` liefert alle Daten zu einem einzelnen Artikel. Sie wird folgendermaßen verwendet:
 - auf der öffentlichen Artikelseite, auf der sie nur veröffentlichte Artikel anzeigen soll
 - auf der Admin-Seite zum Bearbeiten von Artikeln, dort müssen sowohl veröffentlichte als auch unveröffentlichte Artikel angezeigt werden

Daher hat die Methode zwei Parameter, um die richtigen Daten für diese Seiten zu erhalten:

- `$id` ist die ID des zu erhaltenden Artikels.
- `$published` gibt an, ob der Artikel veröffentlicht sein muss, um zurückgegeben zu werden oder nicht. Der Standardwert ist `true`, sodass die Methode nur mit dem Wert `false` auf den Admin-Seiten aufgerufen wird.

2. `$sql` enthält die SQL-Abfrage zum Abrufen der Artikeldaten.

3. Eine `if`-Anweisung prüft, ob der Parameter `$published` den Wert `true` hat.
4. Wenn ja, wird eine Suchbedingung zu der in `$sql` gespeicherten SQL-Abfrage hinzugefügt, die besagt, dass der Artikel veröffentlicht sein muss.
5. Der SQL-Code wird ausgeführt, und die Daten über den Artikel werden als Array zurückgegeben.

6. `getAll()` ruft Artikelzusammenfassungen für vier verschiedene Seiten ab:

- Die Homepage zeigt die letzten 6 Artikel.
- Die Kategoriseite zeigt alle Artikel einer Kategorie.
- Die Mitgliederseite zeigt Artikel von einem Mitglied.
- Die Admin-Seite listet alle Artikel zum Bearbeiten oder Löschen auf.

Um dies zu erreichen, benötigt `getAll()` vier Parameter:

- `$published` legt fest, ob Artikel veröffentlicht werden müssen oder nicht. Der Standardwert ist `true`.
 - `$category` ist die ID der Kategorie, aus der die Artikel abgerufen werden sollen. Der Standardwert ist `null`.
 - `$member` enthält die ID des Mitglieds, von dem die Artikel geschrieben werden sollen. Der Standardwert ist `null`.
 - `$limit` ist die maximale Anzahl von Ergebnissen, die der Ergebnismenge hinzugefügt werden sollen. Der Standardwert ist 1000.
7. `$arguments` enthält ein Array von Argumenten, die die Platzhalter in der SQL-Abfrage ersetzen. Die `category`- und `member`-IDs werden wiederholt, da dieselbe Platzhalter nicht zweimal verwendet werden kann (siehe S. 472).
 8. `$sql` enthält die SQL-Abfrage zum Abrufen der Zusammenfassungsdaten.
 9. Die Parameter `$category` und `$member` sind optional, sodass die SQL-WHERE-Klausel für sie zwei Optionen enthält, die beide in Klammern gesetzt werden:

a. `.category_id = :category` fügt den Artikel zur Ergebnismenge hinzu, wenn der Wert in der Spalte `category_id` mit dem im Parameter `$category` festgelegten Wert übereinstimmt. OR `:category1 is null` fügt den Artikel in die Ergebnismenge ein, wenn für den Parameter `$category` kein Wert angegeben wurde (und sein Standardwert `null` belassen wurde). Der Vorgang wird für die Mitglieds-ID wiederholt.
 10. Wenn Artikel veröffentlicht werden müssen, wird diese Klausel zur Suchbedingung hinzugefügt (wie in den Schritten 3-4).
 11. Die Ergebnisse werden geordnet, und der Suchbedingung wird eine LIMIT-Klausel hinzugefügt, um die Anzahl der Artikel zu begrenzen, die der Ergebnismenge hinzugefügt werden. (Dies wird auf der Startseite verwendet, wo nur 6 Artikel angezeigt werden sollen.)
 12. Die Abfrage wird ausgeführt, und alle Treffer werden zurückgegeben.

```
① public function get(int $id, bool $published = true)
{
    $sql = "SELECT a.id, a.title, a.summary, a.content, a.created, a.category_id,
              a.member_id, a.published,
              c.name      AS category,
              CONCAT(m.forename, ' ', m.surname) AS author,
              i.id        AS image_id,
              i.file      AS image_file,
              i.alt       AS image_alt
        FROM article    AS a
        JOIN category   AS c ON a.category_id = c.id
        JOIN member     AS m ON a.member_id   = m.id
        LEFT JOIN image AS i ON a.image_id   = i.id
        WHERE a.id = :id ";                                // SQL-Statement
    if ($published) {
        $sql .= "AND a.published = 1";
    }
    return $this->db->runSQL($sql, [$id])->fetch();    // Daten zurückgeben
}

⑥ public function getAll($published = true, $category = null, $member = null,
                       $limit = 1000): array
{
    $arguments['category']  = $category;                  // Kategorie-ID
    $arguments['category1'] = $category;                  // Kategorie-ID
    $arguments['member']    = $member;                    // Autor-ID
    $arguments['member1']   = $member;                    // Autor-ID
    $arguments['limit']     = $limit;                     // Max. Artikel
    $sql = "SELECT a.id, a.title, a.summary, a.category_id,
              a.member_id, a.published,
              c.name      AS category,
              CONCAT(m.forename, ' ', m.surname) AS author,
              i.file      AS image_file,
              i.alt       AS image_alt
        FROM article    AS a
        JOIN category   AS c ON a.category_id = c.id
        JOIN member     AS m ON a.member_id   = m.id
        LEFT JOIN image AS i ON a.image_id   = i.id
        WHERE (a.category_id = :category OR :category1 IS null)
          AND (a.member_id  = :member  OR :member1  IS null) "; // SQL-Statement
    if ($published) {
        $sql .= "AND a.published = 1 ";
    }
    $sql .= "ORDER BY a.id DESC LIMIT :limit;";
    return $this->db->runSQL($sql, $arguments)->fetchAll(); // Daten zurück
}
```

DAS CMS-OBJEKT VERWENDEN

Die Methoden des CMS-Objekts können auf einer Seite mehrfach verwendet werden. Die Objekte Article, Category und Member werden nur erstellt, wenn sie benötigt werden, und alle nutzen dasselbe Database-Objekt.

Die neue Seite `category.php` zeigt Daten für eine einzelne Kategorie an und ist viel kürzer als die Seite `category.php` in Kapitel 12, da sie keine SQL-Aufrufe enthält.

1. Anstatt die Dateien

`database-connection.php` und `functions.php` an den Anfang jeder Seite zu stellen, wird die Datei `bootstrap.php` (vorgestellt auf S. 529) eingebunden. Sie erstellt die Variable `$cms`, die ein CMS-Objekt enthält, das für die Arbeit mit der Datenbank verwendet wird.

2. Die ID der anzuzeigenden Kategorie wird abgefragt.

Wenn keine gültige Ganzzahl angegeben wurde, wird die Datei `page-not-found.php` eingefügt. Sie endet mit dem `exit`-Befehl von PHP, damit die Seite `category.php` nicht weiter ausgeführt wird.

Der Pfad zu dieser Datei wird mithilfe der APP_ROOT-Konstanten erstellt (die in `bootstrap.php` erstellt wurde), um sicherzustellen, dass der Pfad korrekt ist.

3. Diese Anweisung ruft die Daten über eine Kategorie ab:

- Die Variable `$category` wird deklariert; sie speichert das Array mit den Daten zu dieser Kategorie.
- Die Variable `$cms` enthält das CMS-Objekt, das in `bootstrap.php` erstellt wurde.
- Die Methode `getCategory()` des CMS-Objekts gibt ein Category-Objekt zurück. Dieses Objekt verfügt über Methoden zum Abrufen, Erstellen, Aktualisieren oder Löschen von Kategorien in der Datenbank.

`getCategory()` wird auf dieser Seite zum ersten Mal aufgerufen, daher wird ein Category-Objekt erstellt, bevor es zurückgegeben wird.

Die Methode `get()` des Category-Objekts wird aufgerufen, um die Kategoriedaten abzurufen. Sie hat ein Argument: die ID der abzurufenden Kategorie.

4. Wenn keine Kategorie zurückgegeben wurde, wird die Datei `page-not-found.php` eingefügt.

5. Diese eine Zeile PHP-Code ruft die Daten für alle Artikel in der Kategorie ab und speichert sie in der Variablen `$articles`. (Im vorigen Kapitel waren dafür 12 Codezeilen erforderlich.) Die Methode `getArticle()` des CMS-Objekts gibt ein Article-Objekt zurück. Die Methode `getAll()` des Article-Objekts gibt dann die Zusammenfassungen für die Artikel in dieser Kategorie zurück. Diese Methode wird mit zwei Argumenten aufgerufen:

- `true` gibt an, dass nur Artikel aus der Datenbank gezogen werden sollen, die bereits veröffentlicht wurden.
- `$id` enthält die ID der Kategorie, die die abzurufenen Artikel enthält.

6. Alle Kategorien werden abgerufen, um die Navigation zu erstellen. Zuerst wird die `getCategory()`-Methode des CMS-Objekts aufgerufen, um ein Category-Objekt zu erhalten (sie gibt das in Schritt 3 erstellte Category-Objekt zurück). Die Methode `getAll()` ruft dann alle Kategorien ab. (Die Datei `header.php` wurde so geändert, dass eine Kategorie nur angezeigt wird, wenn sie in der Navigation erscheinen soll).

```
<?php
declare(strict_types = 1); // Strikte Typisierung
① include '../src/bootstrap.php'; // Datei einrichten

② [ $id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // ID validieren
    if (!$id) { // Wenn ID kein Integer
        include APP_ROOT . '/public/page-not-found.php'; // Page not found
    }
]

③ $category = $cms->getCategory()->get($id); // Kategoriedaten abrufen
④ [ if (!$category) { // Wenn Kategorie leer
    include APP_ROOT . '/public/page-not-found.php'; // Page not found
}

⑤ $articles = $cms->getArticle()->getAll(true, $id); // Artikel abrufen
⑥ $navigation = $cms->getCategory()->getAll(); // Navigationskategorien abrufen
$section = $category['id']; // Aktuelle Kategorie
$title = $category['name']; // HTML-<title>-Inhalt
getDescription = $category['description']; // Inhalt Metabeschreibung
?>
<?php include APP_ROOT . '/includes/header.php' ?>
<main class="container" id="content">
    <section class="header">
        <h1><?= html_escape($category['name']) ?></h1>
        <p><?= html_escape($category['description']) ?></p>
    </section>
    <section class="grid">
        <?php foreach ($articles as $article) { ?>
            <article class="summary">
                <a href="article.php?id=<?= $article['id'] ?>">
                    ">
                <h2><?= html_escape($article['title']) ?></h2>
                <p><?= html_escape($article['summary']) ?></p>
            </a>
            <p class="credit">
                Posted in <a href="category.php?id=<?= $article['category_id'] ?>">
                    <?= html_escape($article['category']) ?></a>
                by <a href="member.php?id=<?= $article['member_id'] ?>">
                    <?= html_escape($article['author']) ?></a>
                </p>
            </article>
        <?php } ?>
    </section>
</main>
<?php include APP_ROOT . '/includes/footer.php' ?>
```

WIE CODE-REFACTORING FUNKTIONIERT

Beim Refactoring des Codes wurden die SQL-Anweisungen und der Code zu ihrer Ausführung aus allen PHP-Dateien in den Ordner `public` und `admin` entfernt. Sie wurden durch Aufrufe der Methoden des neuen CMS-Objekts ersetzt.

Die Beispiele in diesem Kapitel haben gezeigt, dass der Code zum Abrufen und Ändern von Daten in der Datenbank von den einzelnen Seiten, die Benutzer anfordern, in Klassen verlagert wurde. Sie haben auch gesehen, wie man mit diesen Klassen Objekte erstellt und ihre Methoden aufruft.

In diesem Buch ist nicht genug Platz, um jede Seite zu zeigen, die die Benutzer anfordern können, oder jede Methode in den Klassen `Article` und `Member`, aber Sie können alle Änderungen im Code-Download überprüfen. Sie können die Seiten im Ordner `c13` mit den Dateien im Ordner `public` im Ordner `c14` vergleichen, um festzustellen, wie die Methoden der neuen Objekte aufgerufen werden. Mit dem Refactoring-Prozess wurden drei Ziele erreicht, um den Code einfacher zu machen:

- Lesbarkeit und Verständlichkeit
- bessere Organisation
- Erweiterung um neue Funktionen

1. Die Änderungen machen die von den Besuchern angeforderten Seiten einfacher zu lesen und nachzuvollziehen, weil sie

- mit einer einzigen `bootstrap.php`-Datei beginnen (statt mit mehreren Dateien),
- Datenbankdaten in einer einzigen Anweisung abrufen oder ändern, was auch die Codemenge der von den Benutzern angeforderten PHP-Seiten reduziert,
- den für die Arbeit mit der Datenbank erforderlichen Code in den neuen Klassen zusammengefasst.

2. Der Code ist jetzt aus den folgenden Gründen einfacher zu pflegen:

- Wenn sich die SQL-Anweisungen oder der zu ihrer Ausführung benötigte Code ändern sollten, müssen sie nur in den Klassendefinitionen aktualisiert werden (nicht in jeder Seite, die diese Daten anfordert).
- Auf das PDO-Objekt kann nur über die neuen Klassendateien zugegriffen werden. Das bedeutet, dass jeder Code, der auf die Datenbank zugreift, in diesen Klassen enthalten sein muss (er kann nicht über den Rest der Website verteilt sein).
- Wenn eine Website verschoben wird oder das CMS auf einem neuen Server installiert wird, muss nur die Datei `config.php` aktualisiert werden.

3. Die Änderungen erleichtern die Erweiterung der Funktionalität des CMS (Hinzufügen neuer Funktionen):

- Sie nutzen eine konsistente Methode zum Hinzufügen neuer Funktionen zur Website, wobei Methoden der neuen Klassen zum Abrufen oder Ändern der in der Datenbank gespeicherten Daten verwendet werden.
- Sie halten den Datenbankcode getrennt von dem Code, der die Daten anzeigt, und dem Code, der die von den Benutzern eingegebenen Daten verarbeitet.

Der Rest des Buchs zeigt Ihnen, wie Sie die Website um neue Funktionen erweitern können, z.B. die Möglichkeit für Mitglieder, ihre eigene Arbeit zu aktualisieren und Beiträge zu kommentieren.

SELBSTLADENDE KLASSEN

Die neuen Klassendefinitionen werden nur dann in eine Seite aufgenommen, wenn diese versucht, Code dieser Klasse zu verwenden. Dies wird durch die Technik Autoloading erreicht, die mit einer anonymen Funktion implementiert wird.

Stößt der PHP-Interpreter auf eine Anweisung, die ein Objekt über eine Klasse erstellt, und wenn die Klassendefinition nicht in die Seite aufgenommen wurde, können Sie ihn anweisen, eine benutzerdefinierte Funktion aufzurufen. Diese versucht, die Klassendefinition zu laden. Der PHP-Interpreter teilt der Funktion über ein Argument den Namen der zu ladenden Klasse mit.

Die PHP-Funktion `spl_autoload_register()` teilt dem PHP-Interpreter den Namen der Funktion mit, die er aufrufen soll, um zu versuchen, die Klasse zu laden. Sie wird in der Datei `bootstrap.php` aufgerufen, die Sie auf S. 529 kennengelernt haben.

Das automatische Laden von Klassen erspart die Verwendung mehrerer `require`-Befehle zur Aufnahme von Klassendefinitionen auf jeder Seite. Es bedeutet auch, dass die Seite eine Klassendatei nur dann enthält, wenn ein Objekt mit dieser Klasse erstellt wird.

Die Funktion, die Klassen lädt, kann als Argument der Funktion `spl_autoload_register()` definiert werden.

Man spricht von **anonymen Funktionen**, da sie keinen Funktionsnamen nach dem Funktionsschlüsselwort haben (und daher von keinem anderen Code aufgerufen werden können).

HINWEIS: Anonyme Funktionen enden mit einem Semikolon nach der schließenden geschweiften Klammer.

```
spl_autoload_register(function ($class)
{
    ① $path = APP_ROOT . '/src/classes/';
    ② require $path . $class . '.php';
});;
```

Diese anonyme Funktion enthält einen Parameter (`$class`) mit dem Namen der Klasse, die geladen werden soll. Der Klassename wird automatisch vom PHP-Interpreter geliefert, wenn die Funktion aufgerufen wird.

In dieser Funktion muss die Datei, die die Klassendefinition enthält, den gleichen Namen wie die Klasse haben.

Zum Beispiel befindet sich die CMS-Klasse in der Datei `CMS.php` und die Article-Klasse in der Datei `Article.php`.

Die anonyme Funktion enthält zwei Anweisungen:

Die Variable `$path` speichert den Pfad zum Ordner `src/classes` mit den Klassendefinitionen. Die Klassendefinitionsdatei wird mit dem Wert in `$path`, dem Namen der Klasse (der als Argument an die Funktion übergeben wird) und der Erweiterung `.php` eingebunden.

VALIDIERUNGSKLASSE MIT STATISCHEN METHODEN

Die letzte neue Klasse in diesem Kapitel ist die Klasse `Validate`, die den gesamten Validierungscode enthält, der zuvor in der Include-Datei `validate.php` enthalten war.

Alle Funktionen aus der Include-Datei validate.php wurden in eine Klasse Validate verschoben, die sich in der Datei Validate.php im Ordner src/classes befindet. Dies zeigt, wie eine Klasse verwendet werden kann, um eine Reihe verwandter Funktionen zusammenzufassen.

Wenn eine Methode nicht auf Daten zugreifen muss, die in den Objekteigenschaften gespeichert sind, kann sie als statische Methode definiert werden. Dadurch kann diese Methode aufgerufen werden, ohne dass zuvor ein Objekt mit der Klasse erstellt werden muss.

Wenn jede Validierungsfunktion eine statische Methode der Klasse `Validate` wird, verwendet sie:

- das Schlüsselwort `public`, damit sie aus dem Code einer beliebigen Datei aufgerufen werden kann
 - das Schlüsselwort `static`, damit sie aufgerufen werden kann, ohne dass zuvor ein Objekt mit dieser Klasse erstellt wurde
 - `camelCase` für die Methodennamen

public static function isEmail(string \$email): bool
KANN VON KANN OHNE ERSTELLTES OBJEKT
BELIEBIGEM CODE AUFGERUFEN WERDEN
VERWENDET WERDEN

Um eine statische Methode aufzurufen, verwenden Sie den Klassennamen, gefolgt von dem Gültigkeitsoperator `::` (auch bekannt als Doppelpunkt-Operator).

Vor dem Klassennamen steht kein \$-Symbol, weil eine statische Methode der Klassendefinition aufgerufen wird (und nicht ein Objekt, das in einer Variablen gespeichert wurde).

The diagram shows a UML dependency relationship. At the top, the text '::OPERATOR' is aligned with a small square symbol, and 'ARGUMENT' is aligned with a bracket symbol. Below this, the word 'Validate' is followed by a double colon (::), then 'isEmail', then parentheses containing '\$member['email']'. This entire sequence is enclosed in a bracket under the 'KLASSE' label. To the right of the method name, another bracket is labeled 'METHODE'.

1. Die neue Klasse `Validate` wird in der Datei `Validate.php` im Ordner `src/classes` gespeichert.
2. Die Validierungsfunktionen, die in der Datei `validate.php` enthalten waren, werden in die Klasse verschoben (ab diesem Zeitpunkt werden sie als Methoden bezeichnet).

Fügen Sie am Anfang jeder Funktionsdefinition die Schlüsselwörter `public` (damit sie von Code außerhalb der Klassendatei aufgerufen werden kann) und `static` (damit sie aufgerufen werden kann, ohne dass zuvor ein Objekt mit dieser Klasse erstellt wurde) hinzu. Verwenden Sie dann camelCase für den Namen der Methode. Die Anweisungen innerhalb der Methode bleiben gleich.

PHP

c14/src/classes/Validate.php

```

① <?php
class Validate
{
    public static function isNumber($number, $min = 0, $max = 100): bool
    {
        return ($number >= $min and $number <= $max);
    }

②    public static function isText(string $string, int $min = 0, int $max = 1000): bool
    {
        $length = mb_strlen($string);
        return ($length <= $min) or ($length >= $max);
    }
}

```

3. Drei Dateien müssen aktualisiert werden, um die statischen Methoden der Klasse `Validate` zu verwenden:

- `article.php`
- `alt-text-edit.php`
- `category.php`

Diese Seiten müssen die Klasse `Validate` nicht enthalten, da sie automatisch mit der Autoload-Funktion geladen wird, wenn eine Methode der Klasse `Validate` zum ersten Mal aufgerufen wird.

In Kapitel 13 wurden die Validierungsfunktionen unter der Bedingung eines ternären Operators aufgerufen. In diesem Kapitel werden die statischen Methoden der Klasse `Validate` auf die gleiche Weise verwendet. Wo die Funktion aufgerufen wurde, wird sie ersetzt durch:

- Klassenname
- Gültigkeitsoperator `::`
- Name der Methode

Alle Argumente, die der Funktion übergeben wurden, werden der statischen Methode auf genau dieselbe Weise übergeben.

PHP

c14/public/admin/category.php

```

③ $errors['name'] = (Validate::isText($category['name'], 1, 24))
    ? '' : 'Name should be 1-24 characters.'; // Name validieren
$errors['description'] = (Validate::isText($category['description'], 1, 254))
    ? '' : 'Description should be 1-254 characters.'; // Beschreibung validieren

```

ZUSAMMENFASSUNG

REFACTORING & DEPENDENCY INJECTION

- Durch Refactoring wird der Code verbessert, leichter nachvollziehbar, wartbar und erweiterbar, ohne dass sich die Funktionalität ändert.
- Mithilfe von Objekten kann Code gruppiert werden, der eine Reihe verwandter Aufgaben ausführt. Er kann dann von mehreren Seiten verwendet werden, sodass das Duplizieren ähnlichen Codes vermieden wird.
- Wenn eine Variable oder Eigenschaft ein Objekt speichert, enthält sie einen Verweis darauf, wo dieses Objekt im Speicher des PHP-Interpreters abgelegt wurde.
- Dependency Injection stellt sicher, dass eine Funktion oder Methode den zur Ausführung einer Aufgabe benötigten Code enthält, indem die Abhängigkeit als Parameter übergeben wird.
- Das automatische Laden von Klassen erspart das Einfügen von Klassendateien in jede Seite.
- Statische Methoden können aufgerufen werden, ohne dass ein Objekt mit der Klasse erzeugt wird, in der sie definiert wurden.

15

NAMENSRÄUME
&
BIBLIOTHEKEN

Code, der für eine bestimmte Aufgabe geschrieben wurde und der dann weitergegeben wird, damit andere ihn in ihren Projekten ebenfalls verwenden können, wird **Bibliothek** genannt.

In vielen Projekten werden Bibliotheken für bestimmte Aufgaben genutzt.

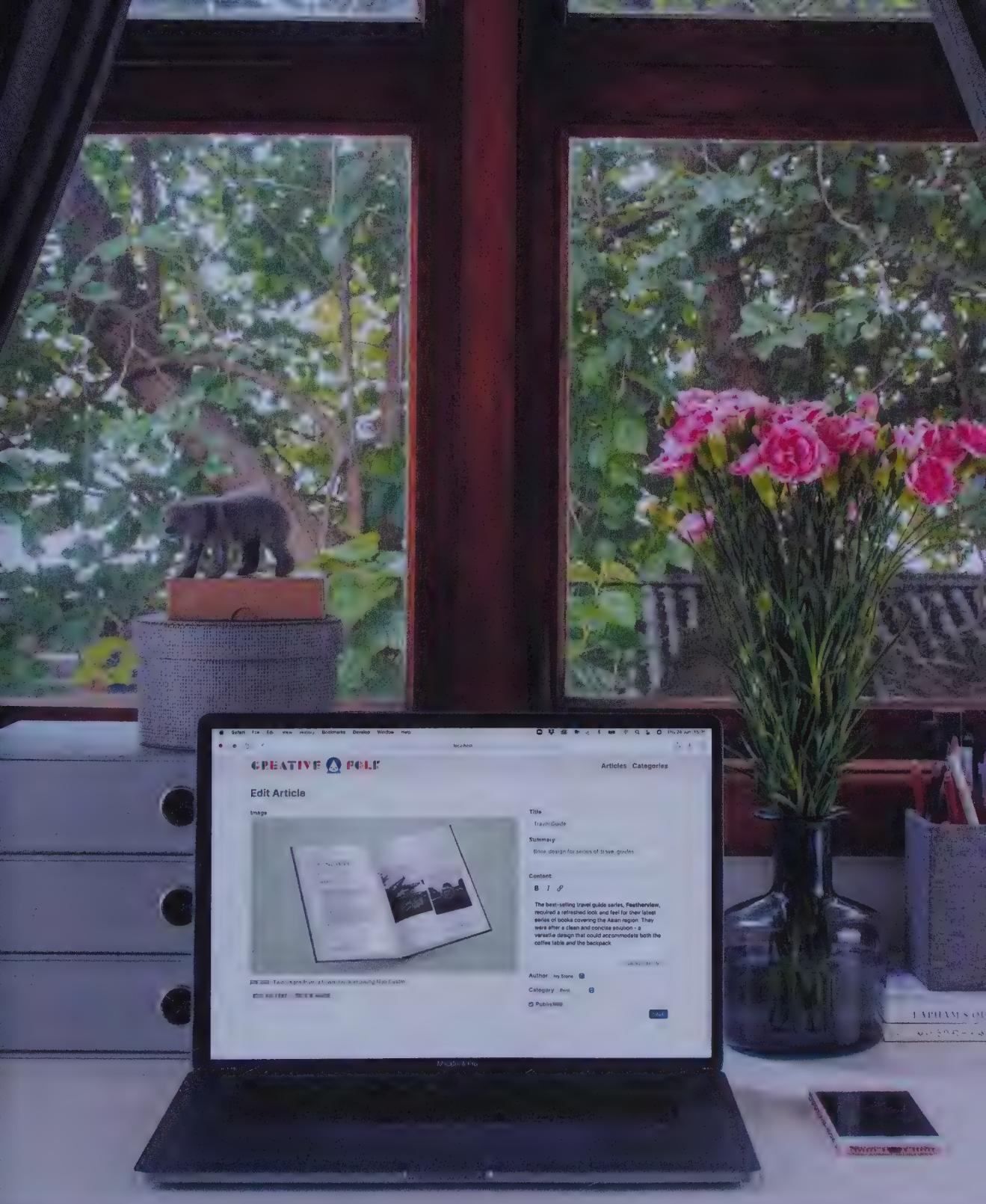
In diesem Kapitel lernen Sie drei populäre PHP-Bibliotheken kennen und sehen, wie sie zur Erweiterung der Funktionalität der Beispiel-Website eingesetzt werden können:

- **HTML Purifier** entfernt unerwünschte HTML-Auszeichnungen aus Texten, die von Benutzern eingegeben wurden. Damit können Benutzer eine begrenzte Anzahl HTML-Tags zu Artikeln hinzufügen.
- **Twig** vereinfacht die Erstellung der HTML-Seiten, die die Besucher sehen. Es wird auf allen Seiten verwendet, die Besucher anfordern können.
- **PHPMailer** erstellt E-Mails und übergibt sie an einen E-Mail-Server, um sie zu versenden. Damit wird eine Kontaktseite erstellt, die eine E-Mail an den Website-Besitzer sendet.

Jede dieser Bibliotheken organisiert den Code mithilfe von Klassen. Sobald ein Objekt mithilfe dieser Klassen erstellt wurde, werden seine Methoden aufgerufen, um Aufgaben auszuführen, für die die Bibliothek konzipiert wurde.

Wenn eine Website eine Bibliothek verwendet, spricht man von **Abhängigkeit** oder **Dependency**, da die Website von dem Code in dieser Bibliothek abhängt. Bevor Sie Bibliotheken verwenden, müssen Sie sich über Folgendes informieren:

- wie PHP **Namensraums** verwendet, damit der PHP-Interpreter in der Lage ist, zwischen zwei oder mehreren Klassen, Funktionen oder Konstanten zu unterscheiden, wenn diese denselben Namen haben
- die Software **Composer**, die Entwicklern hilft, die Bibliotheken zu verwalten, von denen eine Website abhängt (ein sogenannter **Dependency Manager**)
- **Pakete**, also Bibliotheken für die Zusammenarbeit mit Composer



NAMENSRÄUME ERSTELLEN

Namensräume oder **Namespaces** ermöglichen es dem PHP-Interpreter, zwischen zwei Klassen, Funktionen oder Konstanten zu unterscheiden, die den gleichen Namen haben. Namensräume haben Ähnlichkeiten mit Dateipfaden.

Wenn eine Website eine Bibliothek verwendet, kann diese Klassen, Funktionen, Variablen oder Konstanten mit demselben Namen wie eine Klasse, Funktion, Variable oder Konstante in Ihrem Code enthalten. Dies kann eine **Namenskollision** verursachen. Wenn zum Beispiel eine PHP-Datei versucht, zwei Klassendefinitionen mit demselben Namen zu verwenden, und die zweite Klassendefinition in die Seite eingebunden wird, gibt der PHP-Interpreter einen schwerwiegenden Fehler aus, der besagt, dass der Klassename bereits verwendet wird.

Um Namenskollisionen zu vermeiden, werden Bibliotheken in der Regel in einem Namensraum erstellt, der angeibt, dass der gesamte enthaltene Code zu diesem Namensraum gehört. Viele Programmierer erstellen auch einen Namensraum für jede neue Website oder Anwendung.

Um anzugeben, dass der Code zu einem Namensraum gehört, fügen Sie am Anfang der PHP-Datei eine **Namespace-Deklaration** ein. Diese besteht aus dem Schlüsselwort `Namespace` und einem Namensraum. Alle Klassen, Funktionen oder Konstanten, die in der Datei deklariert werden, befinden sich dann in diesem Namensraum.



Namensräume sehen aus wie Dateipfade und bestehen laut den PHP-FIG-Richtlinien (S. 536) aus folgenden Elementen:

- dem oder den Autor(en) des Codes, oft Vendors genannt
- dem Namen der Anwendung oder des Projekts

Um zu verstehen, wie Namensräume funktionieren, sollten Sie sich vor Augen führen, wie Ihr Computer Dateien in Ordnern organisiert.

Ein Ordner kann keine zwei Dateien mit demselben Namen enthalten, wohl aber einen Unterordner mit einer Datei mit demselben Namen. Hier drei Ordner, die alle die Datei `Konten.xlsx` enthalten:

- C:\Dokumente\Konten.xlsx
- C:\Dokumente\Arbeit\Konten.xlsx
- C:\Dokumente\Personal\Konten.xlsx

PHPs eingebaute Klassen, Funktionen und Konstanten sowie alle benutzerdefinierten Klassen, Funktionen, Variablen und Konstanten, denen kein Namensraum zugewiesen wurde, befinden sich im globalen Namensraum. Er ist so etwas wie das Stammverzeichnis eines Computers.

Wird ein Namensraum erstellt, ist das so, als würde ein Ordner innerhalb des globalen Namensraums erstellt; er erlaubt dem PHP-Interpreter, zwischen zwei oder mehr Klassen, Funktionen oder Konstanten mit demselben Namen zu unterscheiden.

Oben sehen Sie den Namensraum für die CMS-Beispielanwendung in diesem Buch; er besteht aus:

- dem Vendor-Namen `PhpBook`
- dem Anwendungs- oder Projektnamen `CMS`

PHP-FIG empfiehlt, Klassennamen und Namensräume in UpperCamelCase zu schreiben. Das bedeutet, dass sie mit einem Großbuchstaben beginnen und, wenn sie mehr als ein Wort enthalten, jedes neue Wort ebenfalls mit einem Großbuchstaben beginnt.

Die benutzerdefinierten Klassen, die im CMS verwendet werden, gehören zu drei verschiedenen Namensräumen. Alle verwenden den Vendor-Namen `PhpBook`, aber der App-/Projektname ändert sich:

- `PhpBook\CMS` wird für den Code verwendet, der die Funktionalität des CMS repräsentiert.
- `PhpBook\Validate` wird für die Validierung verwendet.
- `PhpBook\Email` wird für eine neue Klasse für das Erstellen und Versenden von E-Mails verwendet (siehe S. 598).

Die Klassen im CMS haben den gleichen Projektnamen (und damit den gleichen Namensraum), aber die Klassen `Validate` und `Email` haben eigene Projektnamen (und damit unterschiedliche Namensräume), damit sie in anderen PHP-Projekten (nicht nur im CMS) verwendet werden können. Im Download-Code dieses Kapitels

- wurden Namensräume in die erste Zeile jeder Klassendefinition eingefügt.
- wurden die Klassendefinitionsdateien in Ordner verschoben, die dem Projektnamen des Namensraums entsprechen.

NAMENSRÄUM	DATEIPFAD	ZWECK DER KLASSE
<code>PhpBook\CMS</code>	<code>src\classes\CMS\CMS.php</code>	CMS-Container-Objekt
<code>PhpBook\CMS</code>	<code>src\classes\CMS\Database.php</code>	Zugriff auf die Datenbank über PDO
<code>PhpBook\CMS</code>	<code>src\classes\CMS\Article.php</code>	Artikeldaten abrufen/ändern
<code>PhpBook\CMS</code>	<code>src\classes\CMS\Category.php</code>	Kategoriedaten abrufen/ändern
<code>PhpBook\CMS</code>	<code>src\classes\CMS\Member.php</code>	Mitgliederdaten abrufen/ändern
<code>PhpBook\Email</code>	<code>src\classes\Email\Email.php</code>	E-Mails erstellen und versenden
<code>PhpBook\Validate</code>	<code>src\classes\Validate\Validate.php</code>	Validierungsfunktionen für Formulare

Im letzten Kapitel wurden mit der PHP-Funktion `spl_autoload_register()` in `bootstrap.php` Klassendateien automatisch geladen, wenn sie zur Erstellung von Objekten verwendet wurden.

In diesem Kapitel wurde diese Funktion aus der Datei `bootstrap.php` entfernt. Wie Sie auf S. 571 sehen werden, wird eine andere Technik zum automatischen Laden der Klassendateien verwendet.

CODE IN EINEM NAMENSRAUM VERWENDEN

Um eine Klasse, Funktion oder Konstante in einem Namensraum zu verwenden, geben Sie den Namensraum vor dem Namen der Klasse, Funktion oder Konstanten an. Er wirkt wie ein Präfix.

Befindet sich der von einer PHP-Seite verwendete Code in einem Namensraum, sollte sie einen vollständigen Namensraum verwenden, der aus folgenden Bestandteilen besteht:

- einem Backslash, der den globalen Namensraum bezeichnet (so wie ein Schrägstrich am Anfang eines Dateipfads den Stammordner angibt)
- dem Namensraum für die Klasse
- dem Namen der Klasse, Funktion oder Konstanten

Die folgende Zeile stammt aus `bootstrap.php`; sie erzeugt ein CMS-Objekt unter Verwendung der CMS-Klasse im `PhpBook\CMS`-Namensraum. Wenn der Namensraum nicht wie ein Präfix verwendet würde (wie unten gezeigt), würde der PHP-Interpreter die Klasse, Funktion oder Konstante im globalen Namensraum suchen, statt in dem Namensraum, dem sie hinzugefügt wurde, und würde sie nicht finden.

```
$cms = new \PhpBook\CMS\CMS($dsn, $username, $password);
```

GLOBALER NAMENS- RAUM NAMENS- RAUM KLASSENNAME

Wenn das CMS-Objekt ein Database-Objekt erstellt, kann es mit einem vollqualifizierten Namensraum erstellt werden. Dieser beginnt mit einem `\`, womit der globale Namensraum, der `PhpBook\CMS`-Namensraum und dann der Klassennamen angegeben wird.

Es könnte auch nur der Klassename wie unten gezeigt verwendet werden, da der PHP-Interpreter die Datenbankklasse im gleichen Namensraum wie das CMS-Objekt sucht (und beide sich im Namensraum `PhpBook\CMS` befinden).

```
\PhpBook\CMS\Database($dsn, $username, $password);
```

VOLLQUALI- FIZIERTER NAMENSRAUM KLASSENNAME

```
Database($dsn, $username, $password);
```

KLASSENNAME

Da sich das Database-Objekt im `PhpBook\CMS`-Namensraum befindet, muss ein Backslash `\` vor den Namen aller in PHP integrierten und in dieser Klasse verwendeten Klassen, Funktionen oder Konstanten gesetzt werden, um anzusehen, dass sie sich im globalen Namensraum befinden (siehe rechte Seite). Andernfalls würde der PHP-Interpreter im gleichen Namensraum suchen und sie nicht finden.

Wenn im `PhpBook\CMS`-Namensraum die PDO-Konstanten verwendet werden, um PDO-Optionen festzulegen, oder der Name der `PDOException`-Klasse angegeben wird, um PDO-Ausnahmen in den Artikel- und Kategorieklassen abzufangen, wird ihnen ebenfalls `\` vorangestellt, um dem PHP-Interpreter mitzuteilen, dass sie sich im globalen Namensraum befinden.

NAMENSRÄUME IN CMS-KLASSEN VERWENDEN

Zunächst sehen Sie den Teil der Datei `bootstrap.php`, der das CMS-Objekt für jede Seite erstellt.

PHP

c15/src/bootstrap.php

```
① ...  
② $cms = new \PhpBook\CMS\CMS($dsn, $username, $password);
```

Nachfolgend finden Sie den Anfang der Klasse `Database`. Sie beginnt mit der Namespace-Deklaration. Da die PDO-Klasse nicht im selben Namensraum wie die `Database`-Klasse liegt (sie liegt im globalen Namensraum), wird mit einem Backslash angezeigt, dass die PDO-Klasse im globalen Namensraum liegt.

PHP

c15/src/classes/CMS/Database.php

```
<?php  
① Namesraum PhpBook\CMS;  
  
② class Database extends \PDO  
{  
    protected $pdo = null; // Verweis auf PDO-Objekt speichern  
  
    public function __construct(string $dsn, string $username, string $password,  
        array $options = [])  
    {  
        $default_options[\PDO::ATTR_DEFAULT_FETCH_MODE] = \PDO::FETCH_ASSOC;  
        $default_options[\PDO::ATTR_EMULATE_PREPARES] = false;  
        $default_options[\PDO::ATTR_ERRMODE] = \PDO::ERRMODE_EXCEPTION;  
        $options = array_replace($default_options, $options);  
        parent::__construct($dsn, $username, $password, $options); // PDO-Objekt erstellen  
    }...  
③ }
```

1. Der vollqualifizierte Namensraum für die CMS-Klasse wird vor dem Klassennamen hinzugefügt, um das Objekt zu erstellen.

CODE IN NAMENSRÄUME IMPORTIEREN

Um nicht jedes Mal einen vollqualifizierten Namensraum eingeben zu müssen, wenn Sie eine Klasse verwenden wollen, können Sie die Klasse in den aktuellen Namensraum importieren (den der Rest der Seite verwendet).

Die Klasse `Validate` ist ein Beispiel dafür, dass Sie eine Klasse in einen anderen Namensraum importieren können. Ihre Methoden werden mehrmals aufgerufen, wenn ein Formular validiert wird.

Um diese Methoden aufzurufen, können Sie den voll-qualifizierten Namensraum, den Klassennamen und den Methodennamen wiederholen. (Der Operator :: kennzeichnet eine statische Methode.)

```
\PhpBook\Validate\Validate::isText();
```

Oder Sie können die Klasse `Validate` in den aktuellen Namensraum importieren. Dazu stellen Sie das Schlüsselwort `use`, den Namensraum und den Klassennamen an den Anfang der Datei.

Da die Klasse nun in den aktuellen Namensraum importiert wurde, können Sie ihre Methoden mit dem Klassennamen, gefolgt vom Methodennamen, aufrufen:

use \PhpBook\Validate\Validate;

Wenn es im aktuellen Namensraum bereits eine Klasse `Validate` gäbe, würde dies zu einer Namenskollision führen. Um dies zu umgehen, können Sie einen Alias hinzufügen, d. h. einen Namen, der verwendet werden kann, um auf die importierte Klasse zu verweisen.

Validate::isText();

Der Aliasname kann dann anstelle des Klassennamens verwendet werden, wenn ein Objekt erstellt oder seine Methoden aufgerufen werden. Um einen Alias zu erstellen, fügen Sie das Schlüsselwort `as` hinzu und geben einen Aliasnamen an, der nach dem Import verwendet werden soll:

```
use \PhpBook\Validate\Validate as FormValidate;
```

NAMENSRÄUM	KLASSE	ALIAS
------------	--------	-------

KLASSE IN DEN AKTUELLEN NAMENSRAUM IMPORTIEREN

1. Der Namensraum `PhpBook\Validate` wird am Anfang der `Validate`-Klassendatei hinzugefügt.

2. In der Seite `article.php`, die zum Erstellen und Bearbeiten von Artikeln verwendet wird, importiert die Anweisung `use` die Klasse in den aktuellen Namensraum.

3. Um die Methoden der Klasse `Validate` aufzurufen, wird der Klassennname verwendet, gefolgt vom Methodennamen.

Das Ergebnis ist genau dasselbe wie in der vorherigen Version in Kapitel 14, da der Klassename und seine Methoden in den aktuellen Namensraum (den globalen Namensraum) importiert wurden.

PHP

c15/src/classes/Validate/Validate.php

```
<?php  
① Namensraum PhpBook\Validate;  
  
class Validate  
{  
    ...  
}
```

PHP

c15/public/admin/article.php

```
<?php  
// Teil A: Setup  
declare(strict_types = 1);  
② use PhpBook\Validate\Validate;  
  
include '../../../../../src/bootstrap.php';  
...  
  
// Datenvvalidierung und ggf. Fehlermeldungen  
$errors['title'] = Validate::isText($article['title'], 1, 80)  
    ? '' : 'Title should be 1 - 80 characters.'; // Titel validieren  
$errors['summary'] = Validate::isText($article['summary'], 1, 254)  
    ? '' : 'Summary should be 1 - 254 characters.'; // Zusammenf. validieren  
$errors['content'] = Validate::isText($article['content'], 1, 100000)  
    ? '' : 'Content should be 1 - 100,000 characters.'; // Inhalt validieren  
$errors['member'] = Validate::isMemberId($article['member_id'], $authors)  
    ? '' : 'Not a valid author.'; // Autor validieren  
$errors['category'] = Validate::isCategoryId($article['category_id'], $categories)  
    ? '' : 'Not a valid category.'; // Kategorie validieren
```

SO NUTZEN SIE BIBLIOTHEKEN

Mit Bibliotheken können Sie Code verwenden, den Sie oder andere Programmierer bereits geschrieben haben. Das Tool Composer hilft Ihnen bei der Verwaltung der Bibliotheken, die die Website zum Ausführen benötigt.

Die Verwendung von Bibliotheken, die andere Programmierer geschrieben haben, erspart es Ihnen, für dieselbe Aufgabe Code von Grund auf neu zu schreiben.

Viele Bibliotheken stellen eine oder mehrere Klassen zur Verfügung, die zur Erstellung von Objekten verwendet werden, die die von der Bibliothek angebotene Funktionalität repräsentieren (ähnlich wie die in PHP eingebauten PDO-Klassen die Arbeit mit einer Datenbank ermöglichen oder die in PHP eingebaute DateTime-Klasse Datums- und Zeitangaben repräsentiert und allgemeine Aufgaben damit ausführt).

Wenn Sie eine Bibliothek benutzen, müssen Sie sich nur selten darüber informieren, wie der darin enthaltene PHP-Code seine Aufgabe erfüllt, denn Sie müssen nur lernen

- welche Möglichkeiten die Bibliothek Ihnen bietet
- wie Sie die Bibliothek in die gewünschten Seiten einbinden
- wie man die Objekte erstellen, die die von der Bibliothek angebotenen Funktionen implementieren
- wie Sie Methoden aufrufen oder Eigenschaften der Objekte setzen, um die gewünschte(n) Aufgabe(n) zu erledigen

Wie die meiste Software können auch Bibliotheken regelmäßig aktualisiert (oder sogar komplett neu geschrieben) werden. In jeder Version können neue Funktionen hinzukommen oder Fehler behoben werden, die nach der Freigabe der Bibliothek gefunden wurden.

Wenn Bibliotheken aktualisiert werden, erhalten sie neue Versionsnummern:

- In Hauptversionen werden ganze Zahlen wie v1, v2, v3 usw. verwendet. Bei größeren Aktualisierungen muss möglicherweise der Code in einer PHP-Seite, die die Bibliothek verwendet, geändert werden.
- Minor-Updates oder Point-Releases enthalten in der Regel kleinere Aktualisierungen und Fehlerbehebungen. Sie werden durch einen Punkt und eine zweite oder dritte Nummer gekennzeichnet: v2.1, v2.1.1, v2.1.2, v2.3. Es ist weniger wahrscheinlich, dass sie die Verwendung einer Bibliothek beeinflussen.

Sie als Entwickler müssen alle Bibliotheken, die in einer Website verwendet werden, sorgfältig verwalten:

- Wird die falsche Version verwendet, kann die Website eventuell nicht mehr funktionieren.
- Wurden in der Bibliothek Fehler oder Sicherheitsrisiken gefunden, muss sie aktualisiert werden (andernfalls wird die Website ebenfalls von diesen Fehlern und Sicherheitslücken betroffen sein).

Früher war es eine komplexe Aufgabe, sicherzustellen, dass eine Website die erforderlichen Bibliotheken in der richtigen Version installiert hat, aber das Tool Composer erleichtert diesen Prozess.

HINWEIS: Einige Bibliotheken verwenden Code aus anderen Bibliotheken, sodass sie von der Installation anderer Bibliotheken abhängen.

COMPOSER UND PAKETE VERWENDEN

Bibliotheken für das Zusammenspiel mit Composer werden Pakete genannt. Auf der Website Packagist.org werden die Pakete aufgelistet, die Composer verwenden kann.

Sie können die kostenlose Software Composer auf Ihren Computern ausführen, um Bibliotheken zu verwalten und anzugeben, mit welcher Version der jeweiligen Bibliothek eine Website arbeiten soll.

Damit eine Bibliothek mit Composer funktioniert, muss der Autor der Bibliothek den gesamten Code der Bibliothek in einen einzigen Ordner packen und diesem die Datei `composer.json` hinzufügen. Diese enthält Informationen, die Composer über die Bibliothek und ihre aktuelle Version informieren.

Der Ordner mit der Bibliothek und die Datei `composer.json` werden als **Paket** bezeichnet.

Die Person, die die Bibliothek erstellt hat, kann das Paket dann auf der Website <http://packagist.org> auflisten. Diese ist eine Art Suchmaschine, über die Sie nach nützlichen Bibliotheken suchen können. Die Packagist-Website ist ein sogenanntes Paket-Repository.

Wenn eine neue Version der Bibliothek veröffentlicht wird, aktualisiert der Autor die Datei `composer.json`, sodass Composer erkennen kann, dass dieses Paket eine andere Version der Bibliothek enthält. Dann aktualisiert er die Packagist-Website, um anzugeben, dass eine neue Version verfügbar ist.

HINWEIS: Die in diesem Kapitel verwendeten Bibliotheken, sind bereits im Download-Code für dieses Kapitel enthalten, damit die Beispiel-Website funktioniert.

Wenn Sie Composer verwenden, um die für die Site benötigten Bibliotheken oder aktualisierte Versionen der Bibliotheken herunterzuladen, können Sie:

- das Paket herunterladen, das die Bibliothek enthält. Das Paket ist nicht auf Packagist gespeichert; es wird in der Regel von einer für das Hosten von Quellcode vorgesehenen Website heruntergeladen, etwa GitHub oder Bitbucket.
- andere Bibliotheken herunterladen, die für das Paket benötigt werden (falls sie nicht bereits installiert sind).
- eine Reihe von Textdateien zum Stammordner der Site hinzufügen; diese Dateien sollen helfen, die benötigten Pakete und die benötigten Bibliotheken im Blick zu halten.

Im vorherigen Kapitel haben Sie gesehen, wie die PHP-Funktion `spl_autoload_register()` das automatische Laden von Klassendateien ermöglicht. Dadurch muss die Klassendefinition nicht mehr manuell in die Seite eingefügt werden, damit die Klasse zur Erstellung eines Objekts verwendet werden kann. Wie Sie auf Seite 571 sehen werden, kann Composer eine Datei erstellen, die die Klassendefinitionen für alle mit Composer installierten Pakete automatisch lädt.

Der Composer ist in PHP geschrieben. Er verwendet den Webserver auf dem Entwicklungscomputer, um die Pakete anzufordern und die Textdateien zu erstellen, die die von der Website benötigten Paketversionen aufzeichnen.

PACKAGIST: EIN PAKETVERZEICHNIS

Die Website Packagist listet die Pakete auf, mit denen Composer arbeiten kann. Sie hilft Ihnen, Pakete zu finden, die Sie in Ihren eigenen Projekten verwenden können.

Packagist funktioniert wie eine Suchmaschine. Sie geben den Namen eines Pakets oder einen Begriff ein, der mit der Aufgabe verbunden ist, die Sie durchführen möchten (z.B. Validierung), und Packagist zeigt eine Liste mit Paketen an, deren Name oder Beschreibung mit diesem Begriff übereinstimmt.

Auf der rechten Seite sehen Sie die Seite für die Bibliothek HTML Purifier. Diese Seite zeigt:

1. den Namen des Pakets
2. eine Anleitung zur Installation des Pakets
3. Informationen darüber, was das Paket tut
4. wie oft es bereits installiert wurde
5. bekannte Probleme und Bugs
6. die letzte Version des Pakets
7. das Veröffentlichungsdatum
8. frühere Versionen des Pakets

Bevor Sie sich für ein Paket entscheiden, sollten Sie überprüfen, ob es regelmäßig gewartet wird. Dazu schauen Sie auf Packagist nach:

- wann die Bibliothek zuletzt aktualisiert wurde
- wie viele Versionen der Bibliothek es gibt
- wie viele offene (ungelöste) Probleme es gibt

Wenn es viele ungelöste Probleme gibt oder die Bibliothek in letzter Zeit nicht aktualisiert wurde, hat der Entwickler die Arbeit an der Bibliothek möglicherweise eingestellt.

Dies ist ein Risiko, wenn eine Website für eine bestimmte Aufgabe ein Paket verwendet, statt ihren eigenen Code zu verwenden.

The screenshot shows the Packagist website interface. At the top, there's a search bar with the placeholder "Search packages...". Below it, the package details for "ezyang/htmlpurifier" are displayed. A red circle with the number 1 highlights the package name. A red circle with the number 2 highlights the GitHub link "composer require ezyang/htmlpurifier". A red circle with the number 3 highlights the description "Standards compliant HTML filter written in PHP". Below this, there's a section for "Maintainers" with two profile pictures. Under "Details", there are links for "github.com/ezyang/htmlpurifier", "Homepage", "Source", and "Issues". A red circle with the number 4 highlights the "Installs" count of 32,678,633. A red circle with the number 5 highlights the "Open Issues" count of 60. Below these, a red circle with the number 6 highlights the version "v4.13.0". To the right, a timestamp "2020-06-29 00:56 UTC" and a red circle with the number 7 are shown. Further down, under "requires", there's a dependency on "php: >=5.2". Under "requires (dev)", there's a dependency on "simpleserializer: dev-master@#72de02a7b80c6bb8864ef9bf66d41d2f58fb26bd". At the bottom, there's a license notice "LGPL-2.1-or-later" and a copyright notice "© Edward Z. Yang". A red circle with the number 8 highlights the "dev-master" branch, which has a commit count of 3,000. Below this, a list of other versions is shown: v4.12.0, v4.11.0, v4.10.0, v4.9.3, v4.9.2, and v4.9.1.

COMPOSER UND PAKETE INSTALLIEREN

Composer muss auf Ihrem Entwicklungsrechner installiert werden. Er verfügt über keine grafische Benutzeroberfläche, sondern wird über die Befehlszeile ausgeführt.

Um Composer zu installieren, besuchen Sie die Composer-Website:

<https://getcomposer.org/download/>

Wenn Sie Hilfe bei der Installation von Composer benötigen, sehen Sie hier nach:

http://notes.re/installing_composer

Sobald Composer installiert ist, öffnen Sie das Terminal (Mac) oder die Befehlszeile (Windows) auf Ihrem Computer und navigieren zum Stammverzeichnis Ihrer Website. Falls Sie mit der Bedienung nicht vertraut sind, finden Sie hier eine grundlegende Anleitung:
<http://notes.re/command-line>

Wenn Sie zum Stammordner der Website navigiert sind, geben Sie das Wort Composer in die Befehlszeile ein und drücken die Enter- oder Return-Taste. Sie erhalten eine Liste der Optionen und Befehle, die Composer akzeptiert.

```
Last login: Wed Nov 6 18:11:54 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208056.
-Jon-MacBook-Pro:~ Jon$ composer
Composer
Composer version 1.7.5 2019-11-01 17:20:57
Usage: composer [options] [arguments]
  -V, --version          Display this help message
  -q, --quiet            Do not output any message
  -v, --verbose           Display this application version
  -o, --output            Force API output
  --ansi                 Disable ANSI output
  -n, --no-interaction   Do not ask any interactive question
  -profile               Display timing and memory usage information
  -no-plugins            Whether to disable plugins.
  -d, --working-dir      If specified, use the given directory as working directory.
  --prefer-source         Prevent use of the cache
  --prefer-dist           Increase the verbosity of messages: 1 for normal output, 2 for more
  --ansi                 Shows the short information about Composer.
  -A, --ansi              Creates an archive of the current composer.lock file.
  -C, --create-project    Opens package repository URL or homepage in your browser.
  -C, --create-project=dir Checks that platform requirements are satisfied.
  -C, --create-project=dir Clears composer's internal package cache.
  -C, --create-project=dir Clears composer's internal package cache.
  -C, --create-project=dir Sets config options.
  -C, --create-project=dir Creates new project from a package into given directory.
  -C, --create-project=dir Shows which packages cause the given package to be installed.
  -C, --create-project=dir Diagnoses the system to identify common errors.
  -C, --create-project=dir
```

Quelle: Packagist

Um ein Paket in einem Projekt zu verwenden, suchen Sie die Seite für das Paket auf der Packagist-Website. Paketnamen bestehen aus zwei Teilen: dem Autor des Pakets und dem Projektnamen (beides kann identisch sein), getrennt durch einen Schrägstrich. Der folgende Paketname gehört zum Paket HTML Purifier.

ezyang/htmlpurifier

AUTOR

PROJEKT

Als Nächstes suchen Sie die Anleitung, die das Paket installiert.

Diese wird unter dem Paketnamen auf Packagist angezeigt (Schritt 2 auf der linken Seite). Um Composer mitzuteilen, dass eine Site auf HTML Purifier angewiesen ist, verwenden Sie den folgenden Befehl:

composer require ezyang/htmlpurifier

- composer weist den Computer an, Composer auszuführen.
- require gibt an, dass das Projekt ein Paket benötigt.
- Der Paketname gibt das Paket an, das für dieses Projekt heruntergeladen werden soll.

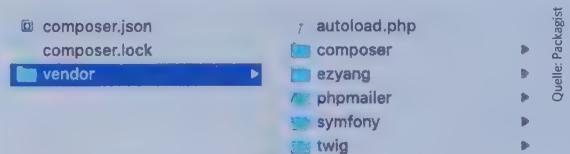
Nachdem Sie die Befehlszeile geöffnet, zum Stammverzeichnis Ihrer Website navigiert sind und die Anweisungen von Packagist eingegeben haben, drücken Sie die Eingabetaste. Composer lädt die neueste Version des Pakets in einen Ordner im Stammverzeichnis der Website herunter (siehe nächste Seite).

Wird für ein Projekt mehr als ein Paket benötigt, wiederholen Sie die obigen Schritte für jedes einzelne Paket.

PAKETE MIT COMPOSER VERWALTEN

Wenn mit Composer ein Paket installiert wird, von dem eine Website abhängt, erstellt er eine Reihe von Dateien und Ordnern im Stammverzeichnis, die ihm helfen, die Versionen des Pakets zu verwalten.

Wird mit dem Befehl `require` das erste Paket für eine Website erstellt, fügt Composer eine Reihe von Dateien und Ordnern in das Stammverzeichnis der Website ein. Sie sind im Screenshot rechts dargestellt und in der Tabelle unten beschrieben. Werden weitere Pakete installiert, aktualisiert Composer diese Dateien und Ordner.



DATEI/ORDNER	ZWECK
<code>composer.json</code>	Eine Datei mit Details zu den Paketen, die für dieses PHP-Projekt benötigt werden
<code>composer.lock</code>	Eine Datei mit Daten zu den Paketversionen und dem Ort, von dem sie heruntergeladen wurden
<code>vendor/</code>	Ein Ordner im Stammverzeichnis, in dem die Pakete gespeichert werden
<code>vendor/autoload.php</code>	Eine Datei, die in die Seiten eingefügt wird, damit die Klassen für die Pakete automatisch geladen werden
<code>vendor/composer/</code>	Ein Ordner mit Dateien, die Composer zur Implementierung des automatischen Ladens von Klassen verwendet

Um alle von der Site verwendeten Pakete zu aktualisieren, öffnen Sie die Kommandozeile, navigieren zum Stammverzeichnis des Projekts und geben den folgenden Befehl ein:

`composer update`

- `composer` weist den Computer an, Composer zu starten.
- Mit `update` aktualisiert Composer die Pakete.

Composer sucht nach den neuesten Versionen aller derzeit installierten Pakete und ersetzt die vorhandenen Pakete durch neuere. Er aktualisiert auch die von ihm erstellten Dateien (einschließlich der Datei `composer.lock`, in der die Version des von der Website verwendeten Pakets gespeichert ist). Sobald alle Pakete aktualisiert wurden, müssen Sie die Site gründlich testen, bevor Sie sie in Betrieb nehmen, da Aktualisierungen manchmal die Site beschädigen können.

Um immer ein Paket auf einmal zu aktualisieren, geben Sie den Paketnamen nach der Aktualisierungsanweisung an. Die folgende Anweisung würde nur das HTML-Purifier-Paket aktualisieren:

`composer update ezyang/htmlpurifier`

Wenn eine Site ein Paket nicht mehr benötigt, können Sie den Befehl `remove` verwenden, gefolgt von dem Namen des Pakets, das nicht mehr verwendet wird. Dadurch werden die Paketdateien aus dem `vendor`-Verzeichnis entfernt und die anderen von Composer erstellten Dateien aktualisiert:

`composer remove ezyang/htmlpurifier`

Wenn ein Paket Code aus einem anderen Paket benötigt, lädt Composer auch dieses herunter. Das Twig-Paket, das Sie auf Seite 576 kennenlernen, benötigt zum Beispiel die Pakete im `symfony`-Ordner, die zur gleichen Zeit wie Twig heruntergeladen wurden.

Composer generiert die Datei `autoload.php`, um die Klassen jedes installierten Pakets automatisch zu laden. Sie kann bearbeitet werden, um auch die benutzerdefinierten Klassen im Ordner `src/classes` automatisch zu laden.

Kapitel 14 hat Sie über Autoloading informiert. Composer erstellt die Datei `autoload.php`, um das automatische Laden von Klassen in den von ihm installierten Paketen zu verwalten. Diese Datei wurde in die Datei `bootstrap.php` aufgenommen:

```
require APP_ROOT . '/vendor/autoload.php';
```

Sie können auch manuell Code zu `composer.json` hinzufügen, um Composer anzusegnen, Ihre benutzerdefinierten Klassen automatisch zu laden. Dadurch wird der Aufruf von `spl_autoload_register()` durch die anonyme Funktion in `bootstrap.php` ersetzt. Der graue Code stammt aus der `composer.json`-Datei, die Composer erstellt hat, als die Pakete in diesem Kapitel hinzugefügt wurden. Der grüne Code wurde hinzugefügt, um den Composer anzusegnen, die benutzerdefinierten Klassen automatisch zu laden.

```
{  
    "require": {  
        "ezyang/htmlpurifier": "^4.12",  
        "twig/twig": "^3.0",  
        "phpmailer/phpmailer": "^6.1"  
    },  
    "autoload": {  
        "psr-4": {  
            "PnpBook\\": "src/classes/"  
        }  
    }  
}
```

Nach der Änderung der `composer.json`-Datei müssen Sie zum Stammverzeichnis des Projekts navigieren

und den folgenden Befehl eingeben, um den Autoloader neu zu erstellen:

```
composer dump-autoload
```

`composer.json` ist in JavaScript Object Notation (JSON) geschrieben. Um Ihre eigenen Klassen mit dieser Technik automatisch zu laden, müssen sie einer Reihe von Richtlinien namens PSR-4 folgen, die von der PHP-FIG-Gruppe erstellt wurden. Die Klassen in diesem Abschnitt folgen bereits diesen Richtlinien.

Damit die Beispiele in diesem Kapitel sofort ausgeführt werden können, musste der Download die erforderlichen Pakete im `vendor`-Ordner enthalten. (Jeder Kapitelordner enthält auch eine `composer.json`- und eine `composer.lock`-Datei). So sehen Sie selbst, wie Composer Pakete herunterlädt und die zusätzlichen Dateien und Ordner erstellt:

1. Erstellen Sie einen neuen Ordner.
2. Lassen Sie diesen Ordner geöffnet.
3. Öffnen Sie die Befehlszeile und navigieren Sie zu dem neuen Ordner.
4. Geben Sie den Befehl `composer` ein, um Composer zu starten.
5. Suchen Sie das gewünschte Paket auf Packagist.
6. Geben Sie den Installationsbefehl in die Befehlszeile ein. Zum Beispiel laden diese drei Befehle die in diesem Kapitel verwendeten Pakete:

```
composer install ezyang/htmlpurifier
```

```
composer install twig/twig
```

```
composer install phpmailer/phpmailer
```

Während die einzelnen Pakete heruntergeladen werden, erscheinen die Dateien und Ordner in dem Ordner, den Sie erstellt und in der Befehlszeile angesteuert haben.

HTML-BEREINIGUNG: HTML-INHALTE ZULASSEN

Die Bibliothek HTML Purifier kann Code entfernen, der einen XSS-Angriff verursacht. Besucher können dann Inhalte erstellen, die HTML enthalten, wobei alle potenziell gefährlichen Markups entfernt werden.

Wenn bisher im CMS ein vom Benutzer eingegebener Text auf einer Seite angezeigt wurde, wurde er mit einem Escape-Zeichen versehen, um einen XSS-Angriff zu verhindern (S. 244–247). Dabei wurden die fünf reservierten HTML-Zeichen durch Entitäten ersetzt, was bedeutete, dass Benutzer keine Inhalte mit HTML-Markup erstellen konnten.

In diesem Abschnitt erfahren Sie, wie Sie zulassen können, dass in einem Artikel einige grundlegende HTML-Tags und -Attribute enthalten sind.

Im CMS-Beispiel darf der Text Absätze, fetten und kursiven Text, Links und Bilder enthalten.

Der PHP-Code für das Entfernen von Markierungen, die einen XSS-Angriff verursachen könnten, ist sehr kompliziert. Statt ihn selbst zu schreiben, verwenden Sie deshalb das Paket **HTML Purifier**. Damit erledigen Sie diese Aufgabe in nur zwei Codezeilen.

Das HTML Purifier-Paket befindet sich im `vendor`-Ordner des Code-Downloads für dieses Kapitel und ist als erforderliches Paket in der `composer.json`-Datei aufgeführt. Sein Name auf Packagist ist `ezyang\htmlpurifier`.

Da die Aufgaben, die HTML Purifier zum Entfernen unerwünschten Markups durchführen muss, recht komplex sind, verwenden wir es, um potenziell gefährliches HTML zu entfernen, wenn ein Artikel gespeichert wird (und nicht jedes Mal, wenn er angezeigt wird). Dies spart Server-Ressourcen, da Artikel häufiger angezeigt als erstellt oder bearbeitet werden.

Unten sehen Sie die Seite `article.php` im Verwaltungsbereich, wo Artikel erstellt oder bearbeitet werden. Der Artikelinhalt wird in einem einfachen visuellen Editor geschrieben, der mit der JavaScript-Bibliothek **TinyMCE** erstellt wurde, die das HTML-Element `<textarea>` durch den unten abgebildeten Editor ersetzt.

Content:

B I ⌂

The best-selling travel guide series, **Featherview**, required a refreshed look and feel for their latest series of books covering the Asian region. They were after a clean and concise solution – a versatile design that could accommodate both the coffee table *and* the backpack.

P

POWERED BY TINY

Quelle: Packagist

Wenn das Formular abgeschickt wird, entfernt HTML Purifier Markup, das einen XSS-Angriff verursachen könnte, aus dem Artikelinhalt, bevor er in der Datenbank gespeichert wird.

Wenn der Artikel auf einer Seite angezeigt wird, darf er nicht maskiert werden. Jegliches HTML, das er enthält, kann sicher angezeigt werden.

Die in PHP eingebaute Funktion `strip_tags()` entfernt zwar HTML-Tags aus dem Text, aber nicht immer auch Attribute, sodass sie XSS-Angriffe nicht verhindern kann.

Um Markup zu entfernen, das einen XSS-Angriff verursachen könnte, erstellen Sie zunächst mithilfe der Klasse `HTMLPurifier` ein `HTMLPurifier`-Objekt. Rufen Sie dann dessen `purify()`-Methode auf.

1. Wenn eine PHP-Datei Text mit Markup akzeptiert, erstellen Sie mithilfe der Klasse `HTMLPurifier` ein `HTMLPurifier`-Objekt. Der `HTMLPurifier` hat keinen eigenen Namensraum; er wird in den globalen Namespace geschrieben. Im Folgenden wird das `HTMLPurifier`-Objekt in der Variablen `$purifier` gespeichert.

2. Dann wird die Methode `purify()` von `HTMLPurifier` aufgerufen.

Ihr einziges Argument ist die Zeichenkette, die bereinigt werden soll (weil sie möglicherweise gefährliches Markup enthält). Sie entfernt alle Markierungen, die ein XSS-Risiko repräsentieren, sowie alle Tags oder Attribute, die nicht in XHTML 1.0 enthalten sind, und gibt dann den Text ohne dieses Markup zurück.

```
① $purifier = new HTMLPurifier();
② $text = $purifier->purify($text);
```

Das `HTMLPurifier`-Objekt kennt die Eigenschaft `config`. Sie enthält ein weiteres Objekt, das Optionen konfiguriert, die die Funktionsweise von `HTML Purifier` steuern.

Sie können zum Beispiel angeben, welche Tags und Attribute erlaubt sind, und alle anderen werden entfernt. Sobald das `HTML Purifier`-Objekt erstellt wurde, können Sie die Einstellungen des Konfigurationsobjekts mithilfe seiner `set()`-Methode mit zwei Argumenten ändern:

- der Eigenschaft, die Sie aktualisieren möchten
- den Werten, die verwendet werden sollen

Die Eigenschaft `HTML.Allowed` gibt zum Beispiel an, welche HTML-Tags und -Attribute erlaubt sind.

Tags, die im Text erscheinen dürfen, sollten als kommasseparierte Tag-Liste (ohne spitze Klammern) angegeben werden. Um zum Beispiel die Tags `<p>`, `
`, `<a>` und `` zu erlauben, würde der Wert für die Eigenschaft `HTML.Allowed` so lauten:

`p,br,a,img`

Um Attribute für ein Element zuzulassen, setzen Sie die zulässigen Attributnamen in die eckigen Klammern nach den Elementnamen. Um mehr als ein Attribut für ein Element zuzulassen, trennen Sie die einzelnen Attribute durch ein Pipe-Symbol `|`.

Um zum Beispiel die Attribute `` und `` für die Tags `<a>` und `` zuzulassen, würde der Wert für die Eigenschaft `HTML.Allowed` so lauten:

`p,br,a[href],img[src|alt]`

```
$purifier->config->set('HTML.Allowed', 'p,br,a[href],img[src|alt]');
```

HTML PURIFIER IN DAS CMS EINFÜGEN

Damit die Benutzer dem Artikelinhalt einfaches Markup hinzufügen können, muss die Seite `article.php` im Admin-Bereich aktualisiert werden.

1. Die Seite `article.php` im Admin-Bereich wird zum Erstellen oder Bearbeiten von Artikeln verwendet. Nach dem Absenden des Formulars werden die Artikeldaten aus dem Formular übernommen, wie in Kapitel 13 gezeigt.

2. Sobald der Inhalt abgerufen wurde, wird ein `HTMLPurifier`-Objekt mithilfe der Klasse `HTMLPurifier` erstellt.

Die Klasse `HTMLPurifier` benötigt keinen vollqualifizierten Namensraum, da sie sich im globalen (statt ihrem eigenen) Namensraum befindet.

Die von Composer generierte Autoload-Datei (die in der Datei `bootstrap.php` enthalten ist) sorgt dafür, dass die erforderlichen Klassendefinitionen automatisch in die Seite aufgenommen werden, wenn das Objekt erstellt wird.

3. Die Eigenschaft `HTML.Allowed` des Konfigurationsobjekts von `HTMLPurifier` wird aktualisiert, um festzulegen, welche Tags und Attribute im Markup erscheinen dürfen.

4. Die Methode `purify()` des `HTMLPurifier`-Objekts wird aufgerufen, um alle Tags und Attribute aus dem Artikelinhalt zu entfernen, die nicht in Schritt 3 angegeben wurden.

5. Da der Inhalt nun bereinigt wurde, sollte er bei der Anzeige auf der Seite nicht mehr mit Escapezeichen versehen werden, da kein Risiko eines XSS-Angriffs mehr besteht.

HINWEIS: Die Datei `article.php` im Code-Download enthält kein HTML, da sie Twig-Templates verwendet (siehe S. 576).

Die Verwaltungsseite für den Artikel zeigt einen einfachen visuellen Editor für den Inhalt des Artikels an, wobei Buttons es dem Benutzer ermöglichen, fetten oder kursiven Text und Links hinzuzufügen.

Er wird mit der JavaScript-Bibliothek TinyMCE erstellt. Um die kostenlose Version des Editors nutzen zu können, müssen Sie sich auf der Website des Entwicklers anmelden: <https://tiny.cloud> (wenn Sie kein Konto zur Nutzung des Editors erstellen, wird auf Ihren Webseiten die Meldung angezeigt, dass das Produkt nicht registriert ist). Sobald Sie sich angemeldet haben, gehen Sie in die Datei `templates/admin/layout.html`.

6. Ein HTML-`<script>`-Tag lädt den TinyMCE-Editor. Sie sollten dieses Tag durch das Tag ersetzen, das Sie bei der Anmeldung für TinyMCE erhalten haben, da es einen sogenannten API-Schlüssel enthält. Der API-Schlüssel identifiziert Ihr Konto und wird dort angezeigt, wo `no-api-key` steht.

7. Mit dem Template `layout.html` (vorgestellt auf S. 577) wird das Aussehen aller Verwaltungsseiten gesteuert. Eine JavaScript-if-Anweisung prüft also, ob die aktuelle Seite ein Element mit einem `id`-Attribut enthält, dessen Wert `article-content` ist (die `id` des `<textarea>`-Elements, das den Artikelinhalt enthält).

8. Ist dies der Fall, ersetzt die Funktion `TinyMCE init()` das `<textarea>`-Element durch den Editor.

9. Es gibt viele Einstellungen, die Optionen wie das Aussehen des Editors und die Funktionen oder Schaltflächen in der Symbolleiste steuern. Um mehr zu erfahren, besuchen Sie die TinyMCE-Website.

①

```
...  

if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Formular  

    $article['title']      = $_POST['title']; // title abrufen  

    $article['summary']    = $_POST['summary']; // summary abrufen  

    $article['content']    = $_POST['content']; // content abrufen  

    $article['member_id']  = $_POST['member_id']; // member_id abrufen  

    $article['category_id'] = $_POST['category_id']; // category_id abrufen  

    $article['image_id']   = $article['image_id'] ?? null; // Article id für Bild  

    $article['published'] = (isset($_POST['published'])) ? 1 : 0; // Navigation abrufen  

② $purifier = new HTMLPurifier(); // Purifier erstellen  

③ $purifier->config->set('HTML.Allowed', 'p,br,strong,em,a[href],img[src|alt]'); // Tags  

④ $article['content'] = $purifier->purify($article['content']);  

...!  

<!-- Hinweis: Das Formular wird später in eine andere Datei verschoben -->  

<div class="form-group">  

    <label for="content">Content: </label>  

    <textarea name="content" id="article-content" class="form-control">  

        <?= $article['content'] ?>  

    </textarea>  

    <span class="errors"><?= $errors['content'] ?></span>  

</div>
```

⑤

⑥

```
...  

<script src="https://cdn.tiny.cloud/1/no-api-key/tinymce/5/tinymce.min.js"  

        referrerPolicy="origin"></script>  

<script>  

    if (document.getElementById('article-content')){  

        tinymce.init({  

            menubar: false,  

            selector: '#article-content',  

            toolbar: 'bold italic link',  

            plugins: 'link',  

            target_list: false,  

            link_title: false  

        });  

    }  

</script>  

</body>  

</html>
```

Auf S. 585 und S. 593 sehen Sie, wie das Template `article.html` im öffentlichen Bereich der Website aktualisiert wird, um zu verhindern, dass das Markup im Artikelinhalt (das mit HTML Purifier sicher gemacht wurde) umgangen wird.

TWIG: EINE TEMPLATE-ENGINE

Template-Engines trennen den PHP-Code, der Daten abruft und verarbeitet, von dem Code, der zur Erstellung der an die Browser gesandten HTML-Seiten verwendet wird. In diesem Buch verwenden wir die Template-Engine Twig.

Bislang bestand jede PHP-Seite in diesem Buch aus zwei Teilen:

- Der erste Teil PHP zum Abrufen und Verarbeiten von Daten. In diesem Teil werden die Daten, die dem Besucher angezeigt werden sollen, in Variablen gespeichert, damit sie im zweiten Teil der Seite angezeigt werden können.
- Der zweite Teil erstellt eine HTML-Seite, die an den Besucher gesendet wird. Dabei werden die Werte verwendet, die in den Variablen im ersten Teil der Seite gespeichert wurden.

Wenn eine Website eine Template-Engine verwendet, bleibt der PHP-Code zum Abrufen und Verarbeiten der Daten in derselben Datei.

Der zweite Teil, der den für die Besucher sichtbaren HTML-Code erstellt, wird in eine Reihe von Dateien, sogenannte **Templates**, ausgelagert. Dies bedeutet eine Trennung zwischen

- dem **Anwendungscode**: PHP-Code, der die erforderlichen Aufgaben der Website ausführt
- dem **Präsentationscode**: Code, der die für den Nutzer sichtbaren HTML-Seiten erstellt

Diese Trennung ist besonders bei Websites beliebt, bei denen verschiedene Entwickler für die folgenden Bereiche zuständig sind:

- **Backend**: der PHP-Code, der auf dem Server läuft
- **Frontend**: der Code, der im Browser angezeigt wird

Der Grund dafür ist, dass sie den Code Ihrer Kollegen nicht verstehen müssen, und es daher weniger wahrscheinlich ist, dass Sie ihn unbrauchbar machen.

Twig-Templates verwenden kein PHP, um die Daten anzuzeigen, sondern eine andere Syntax mit einem kleineren Befehlssatz. Viele Leute finden, dass sie für einen Frontend-Entwickler leichter zu erlernen ist als PHP. So könnten Sie zum Beispiel den Inhalt der Variablen \$title in PHP so schreiben:

```
<p><?= htmlspecialchars($title); ?></p>
```

In einem Twig-Template würden Sie schreiben:

```
<p>{{ title }}</p>
```

Die geschweiften Klammern teilen Twig mit, dass der Wert angezeigt werden soll, der in der Variablen title gespeichert ist.

HINWEIS: Variablennamen in Twig beginnen nicht mit einem \$.

Die Verwendung von Twig als Template-Engine verbessert auch die Sicherheit der Website, denn:

- Twig kann automatisch alle reservierten HTML-Zeichen durch Entities ersetzen, um das Risiko eines XSS-Angriffs zu vermeiden. Der Frontend-Entwickler muss sich nicht jedes Mal daran erinnern, htmlspecialchars() zu verwenden, wenn eine Seite einen vom Benutzer erstellten Wert ausgibt.
- Der gesamte PHP-Code in der Template wird ignoriert, sodass kein Risiko besteht, dass jemand versehentlich unsicheren PHP-Code in das Template einfügt.

Templates können Code mithilfe der **Vererbung** gemeinsam nutzen. Diese stellt einen anderen Ansatz dar als Include-Dateien. Sie ist hilfreich, da sich öffnende und schließende Tags in derselben Datei befinden können (sie werden nicht auf mehrere Dateien aufgeteilt).

In den vorherigen Kapiteln wurden die Kopf- und Fußzeilen jeder Seite in zwei Include-Dateien gespeichert. Dies bedeutete, dass sich die öffnenden und schließenden Tags in unterschiedlichen Dateien befanden.

Jede Seite enthielt diese beiden Dateien, und ihr Code wurde dorthin kopiert, wo die include- oder require-Anweisung in der Seite platziert war.

Include: header.php

```
<html>
  <head> ... </head>
  <body>
```

Include: footer.php

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

Page: category.php

```
<?php include 'header.php'; ?>
<h1>Content goes here</h1>
<?php include 'footer.php'; ?>
```

Twig verwendet ein **Parent-Template** für den gesamten Code, der auf jeder Seite angezeigt wird. Es ist leicht zu bearbeiten, da sich die öffnenden und schließenden Tags in einer Datei befinden. Die Parent-Datei enthält Blöcke, die von einem Child-Template überschrieben werden können.

Parent Template: layout.html

```
<html>
  <head> ... </head>
  <body>
    {% block content %}
      <h1>Content goes here</h1>
    {% endblock %}
  </body>
</html>
```

Child-Templates **erweitern** den Code in einem Parent-Template. Sie basieren auf dem Code im Parent-Template und können die Blöcke im Parent-Template überschreiben. Im Folgenden wird der content-Block überschrieben.

Child Template: category.html

```
{% extends 'layout.html' %}

{% block content %}
  <h1>This replaces anything that was
  in the content block in the
  layout.html file.</h1>
{% endblock %}
```

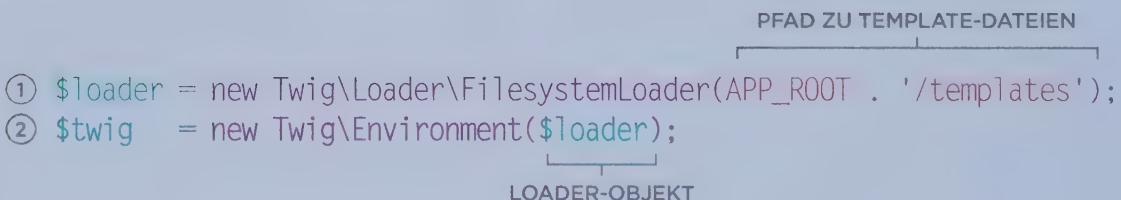
TWIG-OBJEKTE ZUM RENDERN EINES TEMPLATES NUTZEN

Eine Template-Engine fügt die Daten, die eine PHP-Seite in einer Variablen gespeichert hat, dem richtigen Teil des Templates hinzu, um den an den Browser gesandten HTML-Code zu erstellen. Dies wird als Rendern des Templates bezeichnet.

Es gibt mehrere in PHP geschriebene Template-Engines; dieses Buch verwendet Twig. Ihr Packagist-Name ist `Twig\Twig`. Er befindet sich im `vendor`-Ordner des Code-Downloads für dieses Kapitel und ist ein erforderliches Paket in der `composer.json`-Datei im Ordner `c15`.

Jede Seite, die Twig verwendet, muss zwei Objekte aus der Twig-Bibliothek erstellen.

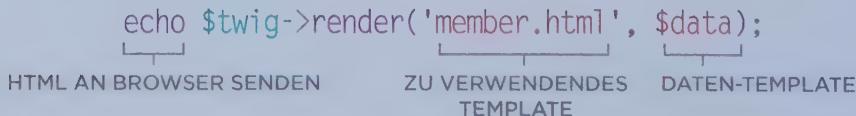
1. Die Klasse `Twig\Loader\FilesystemLoader` erstellt ein Loader-Objekt zum Laden der Template-Dateien. Sie benötigt den Pfad zu dem Ordner mit den Templates.
 2. Die Klasse `Twig\Environment` erstellt ein Twig-Umgebungsobjekt. Sie fügt die Daten in den richtigen Teil des Templates ein. Sie benötigt ein `Twig\Loader`-Objekt, um die Templates zu laden.



Anschließend lädt die `render()`-Methode des Twig-Umgebungsobjekts das Template und erstellt den HTML-Code. Der HTML-Code, den die Methode `render()` zurückgibt, wird mit dem PHP-Befehl `echo` in die Seite geschrieben.

Die Methode `render()` hat zwei Parameter:

- das zur Anzeige der Seite verwendete Template
 - die Daten, die in das Template eingefügt werden



Hinter den Kulissen lädt Twig die Template-Datei und wandelt die Twig-Befehle in PHP-Code um.

Der PHP-Interpreter führt dann diesen PHP-Code aus, um die HTML-Ausgabe zu erstellen, die an den Besucher gesendet wird.

TWIG-OPTIONEN

Wie viele Bibliotheken verfügt auch Twig über Optionen zur Steuerung seiner Funktionsweise. Die Optionen werden in einem Array gespeichert, das bei der Erstellung des Twig-Umgebungsobjekts als Argument angegeben wird (genau wie das PDO-Objekt).

Im Folgenden speichert die Variable \$twig_options ein Array. Es hat zwei Schlüssel beziehungsweise Optionen, die das Verhalten des Twig-Umgebungsobjekts steuern.

```
$twig_options['cache'] = APP_ROOT . '/templates/cache';
$twig_options['debug'] = DEV;

$loader = new Twig\Loader\FilesystemLoader(APP_ROOT . '/templates');
$twig   = new Twig\Environment($loader, $twig_options);
```

LOADER-OBJEKT UMGEBUNGSOPTIONEN

ZWISCHENSPEICHER

Hinter den Kulissen wandelt Twig die Befehle in PHP-Code um, den der PHP-Interpreter ausführt. Der PHP-Code kann **zwischengespeichert** werden (in einer Datei, die auf dem Server gespeichert wird, anstatt bei jedem Aufruf der Seite neu erstellt zu werden). Dadurch werden die Templates schneller geladen und Serverressourcen gespart.

Um den Cache zu aktivieren, benötigt die cache-Option einen absoluten Pfad zu dem Ort, an dem die Dateien gespeichert werden sollen.

Oben wird dieser Pfad in der ersten Option angegeben. Standardmäßig schaltet Twig das Caching nicht ein.

Der Wert für jeden Schlüssel ist eine Einstellung für diese Option. Das Array \$twig_options wird dann als zweites Argument bei der Erstellung eines Twig-Umgebungsobjekts verwendet.

FEHLERSUCHE

Während sich eine Site in der Entwicklung befindet, ermöglicht die Debug-Option, dass Templates Debug-Daten auf den Seiten anzeigen.

Mit der Konstanten DEV, die in der config.php erstellt wurde (siehe S. 528), wird der Wert für die debug-Option festgelegt.

STRIKTE VARIABLEN

Wenn ein Twig-Template eine Variable verwendet, die nicht von der PHP-Seite definiert wurde, erstellt Twig die Variable und gibt ihr den Wert null. Das Twig-Umgebungsobjekt kann so eingestellt werden, dass es eine Ausnahme auslöst, wenn ein Template versucht, eine Variable zu verwenden, die nicht erstellt worden ist. Fügen Sie dazu den Schlüssel strict_variables mit dem Wert true in das Array \$twig_options ein.

GLOBALE VARIABLEN UND ERWEITERUNGEN

Wenn ein Twig-Template PHP-Code enthält, wird dieser nicht ausgeführt. Twig verfügt jedoch über Extensions, die seine Funktionalität erweitern, sowie über globale Variablen, die für alle Templates verfügbar gemacht werden können.

Globale Variablen können verwendet werden, wenn die meisten Templates wahrscheinlich auf einen Wert zugreifen müssen. Die Konstante `DOC_ROOT` wurde beispielsweise in der Datei `config.php` erstellt, damit die PHP-Seiten korrekte Pfade für die Bilder, Stylesheets, Skripte und andere Dateien erstellen können, die der Browser anfordert.

Twig-Templates können nicht auf PHP-Konstanten zugreifen, aber diese Werte können in einer globalen Twig-Variablen gespeichert werden, sodass jedes Twig-Template den Wert verwenden kann. Um eine globale Variable zu erstellen, rufen Sie die Methode `addGlobal()` des Twig-Umgebungsobjekts mit zwei Argumenten auf: dem Namen der Variablen und dem Wert, den sie enthalten soll.

```
$twig->addGlobal('doc_root', DOC_ROOT);
```

GLOBALER VARIABLEN-
NAME WERT

Da Twig-Templates kein PHP verwenden können, können sie die Funktion `var_dump()` nicht nutzen, um die in einer Variablen gespeicherten Werte oder deren Datentyp zu prüfen. Die Twig-Extension `debug` ermöglicht es einem Twig-Template, diese Aufgabe mit der Methode `dump()` zu erfüllen.

Die `debug`-Erweiterung ist als Objekt implementiert. Wenn der Wert der Konstanten `DEV` `true` ist, wird die Methode `addExtension()` des Twig-Umgebungsobjekts aufgerufen, um eine Erweiterung hinzuzufügen, das Argument ist ein neues `DebugExtension`-Objekt.

```
if (DEV){  
    $twig->addExtension(new \Twig\Extension\DebugExtension());  
}
```

NAMENSRÄUM
CLASS

Das einzige Argument für die Twig-Funktion `dump()`, die der PHP-Funktion `var_dump()` ähnelt, ist der Name der Variablen, deren Inhalt Sie sehen möchten.

Die Funktion `dump()` kann den Inhalt einer Variablen nur anzeigen, wenn die `debug`-Option eingeschaltet wurde (siehe vorherige Seite).

BOOTSTRAP ZUM ERSTELLEN VON TWIG-OBJEKten VERWENDEN

Da jede Seite der Website Twig-Templates verwendet, werden der Twig-Loader und die Umgebungsobjekte in der Datei `bootstrap.php` erstellt.

1. Composer hat die Datei `autoload.php` im `vendor`-Ordner erstellt, um automatisch Klassen für die Pakete zu laden, die zur Installation verwendet wurden.
2. Mit der Option `cache` wird Twig angewiesen, die für jedes Template erstellten PHP-Dateien zwischenzuspeichern.
3. Mit der Option `debug` wird der Debug-Modus von Twig aktiviert, wenn die Konstante `DEV` den Wert `true` hat.
4. Es wird ein Twig-Loader-Objekt erstellt. Dieses benötigt den Pfad zu dem Ordner, der die Template-Dateien enthält.

5. Ein Twig-Umgebungsobjekt wird erstellt. Es wird in der Variablen `$twig` gespeichert. Es kann in jeder Seite der Site verwendet werden (wie das CMS-Objekt in der Variablen `$cms`). Die Konstruktormethode benötigt zwei Argumente:

- ein Loader-Objekt, um die Template-Dateien zu laden
 - ein Array von Optionen für das Umgebungsobjekt (das in `$twig_options` gespeichert wurde)
6. Es wird die globale Twig-Variablen `doc_root` hinzugefügt, die den Pfad zum Stammverzeichnis des Dokuments enthält.
7. Eine `if`-Anweisung testet, ob die `DEV`-Konstante wahr ist.
8. Ist dies der Fall, wird die Debug-Erweiterung geladen, sodass die Templates die Funktion `dump()` verwenden können.

PHP

c15/src/bootstrap.php

```
<?php
define("APP_ROOT", dirname(__FILE__, 2)); // Root-Verzeichnis

require APP_ROOT . '/config/config.php'; // Konfiguration
require APP_ROOT . '/src/functions.php'; // Funktionen
① require APP_ROOT . '/vendor/autoload.php'; // Autoload

...
② $twig_options['cache'] = APP_ROOT . '/var/cache'; // Pfad zum Twig-Cache-Ordner
③ $twig_options['debug'] = DEV; // Wenn Dev-Modus, Debug einschalten

④ $loader = new Twig\Loader\FilesystemLoader([APP_ROOT . '/templates']); // Twig-Loader
⑤ $twig = new Twig\Environment($loader, $twig_options); // Twig-Umgebung
⑥ $twig->addGlobal('doc_root', DOC_ROOT); // Document-Root
⑦ if (DEV == true) { // Wenn DEV wahr
    $twig->addExtension(new \Twig\Extension\DebugExtension()); // Debug-Ext. laden
}
```

PHP-SEITEN AKTUALISIEREN

Die von den Besuchern angeforderten PHP-Seiten erhalten Daten aus der Datenbank. Die Daten werden in einem Array gespeichert, das zum Auffüllen der Twig-Templates verwendet wird. Dann wird mit der Methode `render()` der HTML-Code erstellt, den die Besucher sehen sollen.

Die PHP-Seiten, die die Daten abrufen und verarbeiten und dann in Variablen speichern, sind dem ersten Teil der Seiten aus dem vorherigen Kapitel sehr ähnlich.

Der erste Unterschied besteht darin, dass die im Template angezeigten Daten in einem assoziativen Array und nicht in einzelnen Variablen gespeichert werden.

Zum Beispiel ruft die Kategoriseite diese Daten aus der Datenbank ab und speichert sie im Array:

- eine Liste aller Kategorien für die Navigation
- Name und Beschreibung der ausgewählten Kategorie
- zusammenfassende Daten für alle Artikel der Kategorie
- Die ID der Kategorie, die in der Navigation hervorgehoben werden soll

```
INDIZIERTES ARRAY → $data['navigation'] = $cms->getCategory()->getAll();
ASOZIAТИВES ARRAY → $data['category'] = $cms->getCategory()->get($id);
INDIZIERTES ARRAY → $data['articles'] = $cms->getArticle()->getAll(true, $id);
INTEGER → $data['section'] = $category['id'];
```

Sobald die von der Seite benötigten Daten in einem Array gespeichert sind, wird die `render()`-Methode des Twig-Umgebungsobjekts aufgerufen.

Die Methode `render()` fügt die Daten im Array in die Template-Datei ein. Der zurückgegebene HTML-Code wird dann mit dem PHP-Befehl `echo` an den Browser gesendet.

```
echo $twig->render('category.html', $data);
```

SEND HTML TO BROWSER TEMPLATE TO USE DATA TEMPLATE NEEDS

Auf der rechten Seite sehen Sie, wie zwei der PHP-Dateien aktualisiert wurden, um die auf den Seiten angezeigten Daten im Array `$data` zu speichern.

Dadurch werden beide Seiten viel einfacher als zuvor mit dem HTML-Markup.

PHP-DATEIEN ZUM ABRUF UND RENDERN VON DATEN

Die PHP-Dateien im öffentlichen Ordner beginnen mit denselben Anweisungen wie die Versionen in Kapitel 14.

1. Anschließend werden die mit den Methoden des CMS-Objekts aus der Datenbank abgerufenen Daten im Array `$data` gespeichert.

2. Die `render()`-Methode des Twig-Umgebungsobjekts füllt das Template mit den in `$data` gespeicherten Daten.

Der zurückgegebene HTML-Code wird mit dem Befehl `echo` an den Browser gesendet.

PHP

c15/public/index.php

```
<?php
declare(strict_types = 1);
require_once '../src/bootstrap.php';
// Strikte Typisierung
// Datei einrichten

①[$data['articles']] = $cms->getArticle()->getAll(true, null, null, 6); // Zusammenfass.
[$data['navigation']] = $cms->getCategory()->getAll(); // Alle Kategorien

② echo $twig->render('index.html', $data); // Vorlage rendern
```

PHP

c15/public/article.php

```
<?php
declare(strict_types = 1);
require_once '../src/bootstrap.php';
// Strikte Typisierung
// Datei einrichten

$id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); // id validieren
if (!$id) {
    include APP_ROOT . '/public/page-not-found.php'; // Wenn ID nicht valide
} // Page not found
$article = $cms->getArticle()->get($id); // Artikeldaten
if (!$article) {
    include APP_ROOT . '/public/page-not-found.php'; // Wenn article-Array leer
} // Page not found

①[$data['navigation']] = $cms->getCategory()->getAll(); // Kategorien abrufen
[$data['article']] = $article; // Artikel
[$data['section']] = $article['category_id']; // Aktuelle Kategorie

② echo $twig->render('article.html', $data); // Vorlage darstellen
```

AUF DATEN IN TWIG-TEMPLATES ZUGREIFEN

Das Template behandelt jedes Element des `$data`-Arrays als separate Variable, die das Template verwenden kann. Twig-Variablen beginnen nicht mit `$`.

Angenommen, die PHP-Seite erstellt das folgende assoziative Array mit drei Elementen:

```
$data['name'] = 'Ivy Stone';
$data['joined'] = '2021-01-26 12:04:23';
$data['picture'] = 'ivy.jpg';
```

Dann würde das Twig-Template jedes Element des Arrays `$data` als separate Variable behandeln, wobei der Schlüssel im Array zum Variablennamen wird (zur Erinnerung: Variablennamen in Twig beginnen nicht mit einem `$`-Symbol):

- `name`
- `joined`
- `picture`

Hier ist der Wert für eines der Elemente ein anderes Array (und kein ein Einzelwert):

```
$data['category'][‘name’] = ‘Illustration’;
$data[‘category’][‘description’] = ‘Hand-drawn visual storytelling’;
$data[‘category’][‘published’] = true;
```

Das Twig-Template würde dies als die Variable `category` behandeln, ihr Wert wäre ein Array.

Um den Wert für ein Element dieses Arrays zu erhalten, verwenden Sie den Namen der Variablen (`category`), einen Punkt und dann den Namen des Schlüssels (`name`, `description` oder `published`):

- `category.name`
- `category.description`
- `category.published`

Wenn ein Element des Arrays `$data` ein Objekt enthielte, könnte man mit der gleichen Punktsyntax auf Werte zugreifen, die in den Eigenschaften dieses Objekts gespeichert sind.

Wenn ein Template eine nicht im `$data`-Array erstellte Variable nutzt, behandelt Twig sie standardmäßig so, als ob sie erstellt wurde und den Wert `null` hat, anstatt die Fehlermeldung `Undefined variable` zu erzeugen.

Dies kann nützlich sein, da das Template keinen alternativen Wert für Variablen bereitstellen muss, die möglicherweise noch nicht erstellt wurden. Falls erforderlich, kann diese Option jedoch deaktiviert werden.

DATEN IN TWIG-TEMPLATES ANZEIGEN

Twig-Template-Dateien bestehen aus HTML-Tags und Twig-Befehlen. Doppelte geschweifte Klammern `{ { }}` weisen Twig an, den Inhalt auszugeben.

Um einen in einer Variablen gespeicherten Wert anzuzeigen, wird der Variablenname zwischen zwei geschweiften Klammern geschrieben.

```
<h1>{{ category.name }}</h1>
<p>{{ category.description
}}</p>
```

Twig verfügt über eine Reihe von **Filtern**, die mit den in Variablen gespeicherten Daten arbeiten können. Um zum Beispiel zu verhindern, dass Daten maskiert werden, verwenden Sie den Filter `raw`. Dieser Filter wird auf den Artikelinhalt angewendet, der sicher HTML enthalten kann, wenn HTML Purifier verwendet wurde, um unsichere Auszeichnungselemente zu entfernen. Um einen Filter zu verwenden, fügen Sie ein Pipe-Zeichen `|` nach dem Variablennamen und dann den Namen des Filters ein.

```
<p>{{ article.content|raw }}</p>
```

Benötigt ein Filter Daten, um seine Aufgabe zu erfüllen, werden diese in Klammern nach dem Filternamen angegeben (wie eine Funktion).

So kann der `date-filter` ein Datum formatieren.

- Er funktioniert mit denselben Datumsformaten wie die in PHP eingebaute Funktion `strtotime()` (S. 316–317).
- Er formatiert diese Datumsangaben mit denselben Werten wie die in PHP eingebaute Funktion `date()` (S. 316–317).

```
<p>{{ article.created|date('M d Y') }}</p>
```

Da Twig Datumsangaben formatieren kann, wurde die Funktion `format_date()` aus der Datei `functions.php` entfernt.

Wenn die Variable eines der reservierten HTML-Zeichen enthält, maskiert Twig diese automatisch.

Der `e`-Filter maskiert Inhalte, sodass sie auf der Seite sicher angezeigt werden können. Sein Parameter teilt dem Filter mit, wo die Daten verwendet werden sollen. Dies ist wichtig, weil HTML, CSS, JavaScript und URLs verschiedene reservierte Zeichen haben und daher unterschiedliche Zeichen maskiert werden müssen, wenn die Daten in diesen unterschiedlichen Kontexten verwendet werden.

```
<p>{{ article.summary|e('html_attr') }}</p>
```

WERT	KONTEXT
html	HTML-Inhalt
html_attr	Wert der HTML-Attribute
css	CSS
js	JavaScript
url	Text, der Teil einer URL wird

Die Filter `raw`, `date` und `e` sind die einzigen Filter, die im CMS verwendet werden, aber Twig verfügt über weitere Filter zur Formatierung von Zahlen, Zeit und Währung, zum Ändern der Groß- und Kleinschreibung von Text in einer Zeichenkette und zum Sortieren, Verbinden oder Aufteilen von Werten in Arrays.

BEDINGUNGEN IN TWIG-TEMPLATES VERWENDEN

Mit geschweiften Klammern und Prozentzeichen werden öffnende Tags { % und schließende Tags % } erstellt, die Twig mitteilen, wann es eine Aktion wie eine Bedingung oder Schleife ausführen muss.

Eine `if`-Anweisung prüft, ob eine Bedingung den Wert `true` ergibt. Ist dies der Fall, wird der nachfolgende Code ausgeführt, bis ein abschließender `{% endif %}`-Tag vorhanden ist.

Ergibt die Bedingung `false`, wird der Code übersprungen, bis er auf ein abschließendes `{% endif %}`-Tag trifft. Die verwendeten Operatoren sind die gleichen wie in PHP.

```
{% if published == true %}
    <h1>{{ category.name }}</h1>
{% endif %}
```

Wenn die Bedingung nur einen Variablenamen enthält, prüft Twig, ob der Wert in der Variable als `true` behandelt werden würde (nach der Typumwandlung, siehe S. 60–61).

```
{% if published %}
    <h1>{{ category.name }}</h1>
{% endif %}
```

Mehrere Bedingungen können mit `and` und `or` verknüpft werden.

```
{% if time > 6 and time < 12 %}
    <p>Good morning.</p>
{% endif %}
```

Twig unterstützt auch `else`- und `elseif`-Strukturen sowie die ternären und Null-Koaleszenz-Operatoren.

```
{% if time > 6 and time < 12 %}
    <p>Good morning.</p>
{% elseif time >= 12 < 5 %}
    <p>Good afternoon.</p>
{% else %}
    <p>Welcome.</p>
{% endif %}
```

SCHLEIFEN IN TWIG-TEMPLATES VERWENDEN

Twig verfügt über eine `for`-Schleife, die jedes Element in einem Array oder jede Objekteigenschaft abarbeiten kann (wie die `foreach`-Schleife von PHP).

Die `for`-Schleife von Twig ist wie eine `foreach`-Schleife in PHP und wird verwendet, um Elemente in einem Array zu bearbeiten. Die darin enthaltenen Anweisungen werden bei jedem Durchlauf der Schleife wiederholt. Die Schleife wird mit dem Tag `{% endfor %}` abgeschlossen.

```
{% for article in articles %}
    <h2>{{ article.title }}</h2>
    <p>{{ article.summary }}</p>
{% endfor %}
```

Um eine bestimmte Anzahl von Elementen in einer Schleife zu durchlaufen, wird eine etwas andere Syntax verwendet. Das öffnende Tag beginnt mit dem Schlüsselwort `for`.

- Anschließend dient eine Twig-Variable als Zähler, hier `i` (diese Variable kann in der Schleife verwendet werden)
- gefolgt von dem Schlüsselwort `in`
- dann `1..` und eine Variable, die die Anzahl der Durchläufe der Schleife enthält.

```
{% for i in 1..count %}
    <a href="?page={{ i }}">{{ i
}}</a>
{% endfor %}
```

Das öffnende Tag beginnt mit dem Schlüsselwort `for`, dann folgt eine Variable, die das aktuelle Element in der Schleife enthält; danach folgen das Schlüsselwort `in` und die Variable, die das Array oder Objekt enthält, durch das die Schleife laufen soll.

Hätte die Variable `count` den Wert 5, würde die Schleife fünfmal durchlaufen. Beim ersten Durchlauf der Schleife würde die Zählervariable `i` den Wert 1 annehmen, beim nächsten Mal den Wert 2. Dies würde so lange fortgesetzt, bis die Zahl 5 erreicht wäre.

Diese Art von Schleife wird im Such-Template demonstriert, wobei die Ergebnisse mittels Paginierung angezeigt werden.

SO STRUKTURIEREN SIE TEMPLATE-DATEIEN

Ein einziges **Parent**-Template sollte allen Code enthalten, der auf jeder Seite einer Website erscheint. Dieses Parent-Template verwendet **Blöcke**, die von einem **Child**-Template überschrieben werden können.

Wenn Sie für eine Website Twig verwendet, sollte ein **Parent**-Template den Code enthalten, der auf jeder Seite verwendet wird (wie der Code in den Kopf- und Fußzeilendateien in den vorherigen Kapiteln).

Templates können **Blöcke** definieren. Im Parent-Template stellen die Blöcke Abschnitte des Layouts dar, die von anderen Seiten überschrieben werden können (um andere Daten anzuzeigen).

Blöcke beginnen mit einem Tag, das dem Block einen Namen gibt:

```
{% block block-name %}
```

Blöcke enden mit einem schließenden Tag:

```
{% endblock %}
```

Auf dieser Seite sehen Sie das **Parent**-Template **layout.html**. Es enthält den Code, der auf jeder Seite der Webseite erscheint, und besteht aus drei Blöcken:

- **title** zeigt den Text im `<title>`-Tag der Seite an. (Wenn das Child-Template ihn nicht mit einem neuen Wert überschreibt, wird der Text innerhalb dieses Blocks verwendet).
- **content** ist der Bereich, in dem der Hauptinhalt jeder Seite angezeigt wird. Er enthält keinen Standardinhalt, d. h., wenn ein Child-Template keinen content-Block enthält, wird an seiner Stelle nichts angezeigt.
- **footer** enthält die Fußzeile für die Website. Er enthält eine Copyright-Erklärung und das aktuelle Jahr.

Parent Template: layout.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      {% block title %}
        Creative Folk
      {% endblock %}
    </title>
  </head>
  <body>
    {% block content %}{% endblock %}
    <footer>
      {% block footer %}
        &copy; Creative Folk
        {{ 'now'|date('Y') }}
      {% endblock %}
    </footer>
  </body>
</html>
```

Child-Templates können einzelne Seiten der Website repräsentieren. Sie erben den Code vom Parent-Template und stellen Daten bereit, um den Inhalt in den benannten Blöcken des Parent-Templates zu überschreiben.

Child-Template: category.html

```
{% extends 'layout.html' %}

{% block title %}
{{ category.name }}
{% endblock %}

{% block content %}
<h1>{{ category.name }}</h1>
<p>{{ category.description }}</p>

{{ include('article-summaries.
html') }}
{% endblock %}
```

Jeder Seitentyp, den Besucher der Website aufrufen können (Home, Category, Article, Member und Search), hat sein eigenes Child-Template, das das Parent-Template erweitert. Das `extends`-Tag gibt den Namen des zu erweiternden Parent-Templates an:

```
{% extends 'parent-template.html' %}
```

Im Child-Template überschreibt alles zwischen den `block`-Tags den Inhalt des entsprechenden Blocks im Parent-Template.

Auf dieser Seite erweitert das Child-Template `category.html` das `layout.html` und enthält zwei Blöcke:

- `title` ersetzt den Inhalt des `title`-Blocks im Parent-Template.
- `content` ersetzt den Inhaltsblock im Parent-Template.

Das Child-Template enthält keinen `footer`-Block, sodass sich auf der Seite die Copyright-Meldung im `footer`-Block des Parent-Templates befand.

Innerhalb dieses Child-Templates befindet sich im Inhaltsblock eine `include()`-Funktion in geschweiften Klammern, um eine andere Template-Datei einzubinden. Dieses Template zeigt eine Reihe von Artikelzusammenfassungen an:

```
{{ include('article-summaries.html') }}
```

Die übergeordnete Datei, die untergeordnete Datei und die `Include`-Datei können auf die Variablen zugreifen, die mit dem Array `$data` erstellt wurden.

TEMPLATE FÜR ÜBER-GEORDNETE UND UNTER-GEORDNETE KATEGORIEN

Das Child-Template `category.html` zeigt Details zu jeder Kategorie an. Es erbt das gesamte Markup des Parent-Templates `layout.html` auf der rechten Seite.

1. Das `extends`-Tag zeigt an, dass dieses Child-Template den Code von `layout.html` erbt.
2. Der Titelblock in diesem Child-Templates ersetzt den Titelblock im Parent-Template.

Er zeigt den Namen der Kategorie und dann die Worte `on Creative Folk` an.

3. Der `description`-Block im Child-Template ersetzt den `description`-Block im Parent-Template. Er zeigt die Kategoriebeschreibung im `value`-Attribut des `<meta>`-Description-Tags an. Er verwendet den `e()`-Filter, um den Inhalt für die Nutzung in einem Attribut zu maskieren.

4. Der `content`-Block im Child-Template ersetzt den `content`-Block im Parent-Template.

5. Der Name der Kategorie wird in einem `<h1>`-Element angezeigt.

6. Es folgt die Beschreibung der Kategorie, die in einem `<p>`-Element angezeigt wird.

7. Ein `include`-Tag enthält ein Template zur Anzeige von Zusammenfassungen der Artikel in dieser Kategorie.

Das Template `article-summaries.html` (siehe S. 592) durchläuft die Zusammenfassungen im Array `articles` und zeigt die Zusammenfassung für diese Artikel an.

8. Der `content`-Block wird geschlossen.

c15/templates/category.html

Twig

```
①  {% extends 'layout.html' %} 
②  {% block title %}{{ category.name }} on Creative Folk{% endblock %} 
③  {% block description %}{{ category.description|e('html_attr') }}{% endblock %} 

④  {% block content %} 
    <main class="container" id="content"> 
        <section class="header"> 
            <h1>{{ category.name }}</h1> 
        <section class="grid"> 
            {{ include('article-summaries.html') }} 
        </section> 
    </main> 
⑧  {% endblock %}
```

Twig-Templates können beliebige Dateierweiterungen haben. Mit der Erweiterung `.html` erkennen Code-Editoren jedoch, dass die Datei HTML-Code enthält.

Der Code-Editor kann dann den Code hervorheben, als ob es sich um HTML handeln würde, und auch Funktionen wie das Hervorheben von Fehlern anbieten.

Das Parent-Template enthält den auf jeder Seite verwendeten Code.

Es besteht aus drei Blöcken: Titel, Beschreibung und Inhalt.

Auch werden die Kategorien in einer Schleife durchlaufen, um die Hauptnavigation zu erstellen.

TWIG

c15/templates/layout.html

```
<!DOCTYPE html>
<html lang="en-US">
  <head> ...
    <title>{% block title %}Creative Folk{% endblock %}</title>
    <meta name="description" value="{% block description %}Hire creative folk{% endblock %}">
    <link rel="stylesheet" type="text/css" href="{{ doc_root }}css/styles.css"> ...
  </head>
  <body>
    <header>
      <div class="container">
        <a class="skip-link" href="#content">Skip to content</a>
        <div class="logo"><a href="{{ doc_root }}index.php">
          
        </a></div>
        <nav>
          <button id="toggle-navigation" aria-expanded="false">
            <span class="icon-menu"></span><span class="hidden">Menu</span>
          </button>
          <ul id="menu">
            {% for link in navigation %}
            {% if (link.navigation == 1) %}
              <li><a href="{{ doc_root }}category.php?id={{ link.id }}">
                {% if (section == link.id) %} class="on"{% endif %}
                {{ link.name }}</a></li>
            {% endif %}
            {% endfor %}
            <li><a href="{{ doc_root }}search.php">
              <span class="icon-search"></span><span class="search-text">Search</span>
            </li>
          </ul>
        </nav>
      </div>
    </header>
    {% block content %}{% endblock %}
  <footer>
    <div class="container">
      <a href="{{ doc_root }}contact.php">Contact Us</a>
      <span class="copyright">&copy; Creative Folk {{ 'now' | date('Y') }}</span>
    </div>
  </footer>
  <script src="{{ doc_root }}js/site.js"></script>
</body>
</html>
```

TEMPLATE FÜR DIE ARTIKEL-ZUSAMMENFASSUNG

Das Template `article-summaries.html` zeigt Zusammenfassungen mehrerer Artikel an. Sie werden in den Templates für die home-, category-, member- und search-Seiten verwendet.

1. Eine `for`-Schleife durchläuft ein Array von Artikelzusammenfassungen, die in der Variablen `articles` gespeichert sind. In der Schleife wird jeder Artikel in der Variablen `article` gespeichert.

2. Es wird ein Link zu dem Artikel erstellt.

3. Eine Twig-`if`-Anweisung prüft, ob der Artikel ein Bild enthält.
4. Wenn ja, werden das Bild und sein Alt-Text in einem ``-Tag angezeigt.
5. Wenn nicht, folgt auf den Twig-Tag `{% else %}` ein alternativer Code.
6. Ein Platzhalterbild wird angezeigt.
7. Das `{% endif %}`-Tag markiert das Ende der `if`-Anweisung.

8. Der Titel des Artikels wird in einem `<h2>`-Element angezeigt.
9. Die Zusammenfassung wird angezeigt.
10. Ein Link zu der Kategorie, in der sich der Artikel befindet, wird erstellt.
11. Ein Link zur Seite des Artikelautors wird erstellt.
12. Die Schleife endet mit dem Tag `{% endfor %}`.

c15/templates/article-summaries.html

Twig

```
① {# for article in articles #}
<article class="summary">
②   <a href="{{ doc_root }}article.php?id={{ article.id }}">
③     {% if article.image_file %}
        
④     {% else %}
        
⑤     {% endif %}
⑥     <h2>{{ article.title }}</h2>
⑦     <p>{{ article.summary }}</p>
⑧   </a>
⑨   <p class="credit">
⑩     Posted in <a href="{{ doc_root }}category.php?id={{ article.category_id }}">
          {{ article.category }}</a>
⑪     by <a href="{{ doc_root }}member.php?id={{ article.member_id }}">
          {{ article.author }}</a>
⑫   </p>
</article>
⑬ {# endfor #}
```

ARTIKEL-TEMPLATE

1. Dieses Child-Template zeigt einen Artikel. Das extends-Tag weist es an, den Code von layout.html zu erben.
2. Der title-Block zeigt den Titel im <title>-Element an.
3. Der description-Block enthält die Zusammenfassung. Mit dem e-Filter wird der Text für die Verwendung in HTML-Attributen maskiert.

4. Der content-Block zeigt die vollständigen Details des Artikels.
5. Wenn der Artikel ein Bild enthält, wird dieses angezeigt; wenn nicht, wird blank.png angezeigt.
6. Der Titel wird wieder ausgegeben.
7. Der date-Filter formatiert das Datum, an dem der Artikel verfasst wurde.

8. Der Artikelinhalt wird mit dem raw-Filter angezeigt, damit Twig ihn nicht umbricht, da er HTML-Auszeichnungen enthalten kann (die bereits von HTML Purifier für die Anzeige freigegeben wurden).
9. Ein Link zu der Kategorie, in der sich der Artikel befindet, wird erstellt, gefolgt von einem Link zur Seite des Autors.

TWIG

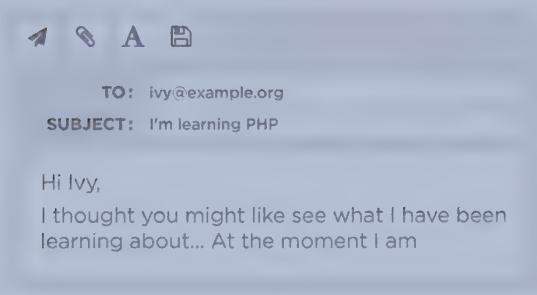
c15/templates/article.html

```
①  {% extends 'layout.html' %}          [           ]  
②  {% block title %}{{ article.title }}{% endblock %}  [           ]  
③  {% block description %}{{ article.summary|e('html_attr') }}{% endblock %}  [           ]  
④  {% block content %}                [           ]  
    <main class="article container" id="content">  
      <section class="image">  
        {% if article.image_file %}  
            
        {% else %}  
            
        {% endif %}  
      </section>  
      <section class="text">  
        <h2>{{ article.title }}</h2>  
        <div class="date">{{ article.created|date('F d, Y') }}</div>  
        <div class="content">{{ article.content|raw }}</div>  
        <p class="credit">  
          Posted in <a href="{{ doc_root }}category.php?id={{ article.category_id }}>  
            {{ article.category }}</a>  
          by <a href="{{ doc_root }}member.php?id={{ article.member_id }}>  
            {{ article.author }}</a></p>  
      </section>  
    </main>  
  {% endblock %}
```

E-MAILS MIT PHPMAILER VERSCHICKEN

Websites senden oft einzelne E-Mails, sogenannte Transaktions-E-Mails. Zum Beispiel kann eine Seite einen Link zum Zurücksetzen des Passworts per E-Mail versenden, oder ein Kontaktformular kann eine Nachricht an den Eigentümer der Website senden.

Wenn Sie eine E-Mail von einem Computer oder mobilen Gerät aus versenden, geben Sie die E-Mail-Adresse des Empfängers, einen Betreff und einen Nachrichtentext an. Das E-Mail-Programm sendet die E-Mail dann an einen SMTP-Server, der die E-Mail an den Empfänger übermittelt.



Wenn ein E-Mail-Programm für die Verwendung einer neuen E-Mail-Adresse eingerichtet wird, muss ihm mitgeteilt werden, wie es sich mit dem SMTP-Server verbinden kann. Normalerweise werden für die Verbindung folgende Angaben benötigt:

- **Hostname** zur Identifikation des SMTP-Servers, so wie ein Domänenname einen Webserver identifiziert
- **Portnummer**, die es verschiedenen Programmen auf demselben Computer ermöglicht, die-selbe Internetverbindung zu nutzen (siehe <http://notes.re/php/ports>)
- **Benutzername und Passwort**, um sich bei einem Konto anzumelden
- **Sicherheitseinstellungen**, um festzulegen, wie der Benutzername und das Passwort sicher übermittelt werden sollen

Wenn eine Website eine E-Mail senden muss, sind dieselben beiden Schritte erforderlich:

- Verbindung zu einem SMTP-Server, der E-Mails versenden kann
- Erstellen der E-Mail und Übergabe an den SMTP-Server

Der Code zur Durchführung dieser beiden Aufgaben ist komplex. Anstatt den Code zum Erstellen und Versenden von E-Mails von Grund auf neu zu schreiben, verwendet die Beispiele die das Paket **PHPMailer**. Dieses wird in vielen beliebten Open-Source-Projekten verwendet, darunter WordPress, Joomla! und Drupal.

Das PHPMailer-Paket befindet sich im vendor-Ordner des Code-Downloads für dieses Kapitel und ist als erforderliches Paket in der Datei composer.json aufgeführt.

Sein Name auf Packagist lautet `phpmailer\phpmailer`.

Obwohl ein Webserver seinen eigenen SMTP-Server betreiben kann, verwenden Websites in der Regel ein spezialisiertes Unternehmen, das einen SMTP-Server für den Versand der E-Mails bereitstellt. Die Gründe:

- Wenn Sie zu viele E-Mails von Ihrem Webserver aus versenden, kann dies dazu führen, dass E-Mail-Dienste Ihren Domänennamen auf eine schwarze Liste setzen und Ihre E-Mails als Spam behandeln.
- Sie haben bessere Erfolgsquoten bei der Zustellung

Eine Liste von Diensten, die Transaktions-E-Mails versenden, finden Sie unter:

<http://notes.re/transactional-emails>.

Sie müssen sich bei einem dieser Dienste anmelden, um den Code für den E-Mail-Versand zu testen.

EINSTELLUNGEN FÜR DEN SMTP-SERVER

Die Angaben zur Verbindung mit dem SMTP-Server sind für jede Website, auf der der CMS-Code ausgeführt wird, unterschiedlich; sie werden daher als Konfigurationsdaten eingestuft und in der Datei config.php gespeichert.

Jede Website, die den CMS-Code verwendet, benötigt andere Einstellungen für die Verbindung zu ihrem SMTP-Server (so wie jede Website andere Details für die Verbindung zu ihrer Datenbank benötigt).

Die Daten für die Verbindung zum SMTP-Server werden als assoziatives Array in der Datei config.php gespeichert, zusammen mit der E-Mail-Adresse des Website-Besitzers.

PHP

```
① $email_config = [  
②     'server'      => 'smtp.YOUR-SERVER.com',  
③     'port'        => 'YOUR-PORT-NUMBER',  
④     'username'    => 'YOUR-USERNAME-HERE',  
⑤     'password'    => 'YOUR-PASSWORD-HERE',  
⑥     'security'    => 'tls',  
⑦     'admin_email' => 'YOUR-EMAIL-HERE',  
⑧     'debug'       => (DEV) ? 2 : 0,  
];
```

Speichern Sie zunächst die Daten für die Verbindung mit dem SMTP-Server:

1. \$email_config ist die Variable, die ein Array von Einstellungen für den Versand von E-Mails enthält.

2. server enthält den Hostnamen des SMTP-Servers.

3. port ist die Portnummer, die der SMTP-Server verwendet.

4. username und password enthalten die Anmelde-daten für das SMTP-Server-Konto.

5. security enthält die Methode, mit der die Daten sicher übertragen werden. Der Wert hierfür ist normalerweise tls, was für Transport Layer Security steht (siehe S. 185).

Fügen Sie dann die beiden anderen benötigten Werte hinzu:

6. admin_email ist die E-Mail-Adresse des Website-Besitzers. An sie werden die Nachrichten des Kontaktformulars gesendet (siehe S. 598–601). Sie ist auch die Absenderadresse für andere E-Mails, die die Website im nächsten Kapitel versendet.

7. debug schaltet Debug-Meldungen ein und aus. Der Wert in der Konstanten DEV bestimmt die Einstellung:

- 2 für die Entwicklung einer Site, um die vom Web-server gesendeten E-Mails und die Antworten des SMTP-Servers anzuseigen
- 0 für eine Live-Site, um Debugging-Meldungen zu deaktivieren (da die Meldungen Daten über das SMTP-Konto anzeigen)

EINE E-MAIL ERSTELLEN UND SENDEN

Erstellen Sie zunächst ein Objekt mit der `PHPMailer`-Klasse und teilen Sie ihm mit, wie es sich mit dem SMTP-Server verbinden soll. Erstellen und versenden Sie dann die E-Mail.

Ein `PHPMailer`-Objekt wird mithilfe der `PHPMailer`-Klasse erstellt, genau wie jedes andere Objekt.

Sein Namensraum ist `PHPMailer\PHPMailer`. Dieser muss bei der Erstellung des Objekts verwendet werden.

Bei der Erstellung des `PHPMailer`-Objekts wird der boolesche Wert `true` als Argument verwendet. Damit wird `PHPMailer` angewiesen, eine Ausnahme zu werfen, wenn ein Problem beim Erstellen oder Senden einer E-Mail auftritt.

```
$phpmailer = new \PHPMailer\PHPMailer\PHPMailer(true);
```

VARIABLE

NAMENRAUM

KLASSE

BEI PROBLEMEN AUSNAHME WERFEN

Nach der Erstellung des `PHPMailer`-Objekts wird das Objekt mit zwei Methoden und acht Eigenschaften so konfiguriert, dass es weiß, wie diese Website E-Mails senden wird.

Sie sind vergleichbar mit den Einstellungen, die ein E-Mail-Programm für ein neues E-Mail-Konto verwendet, und bleiben jedes Mal gleich, wenn die Website eine Transaktions-E-Mail sendet.

EIGENSCHAFT/METHODE	BESCHREIBUNG
<code>isSMTP()</code>	Methode, die angibt, dass ein SMTP-Server für den Versand der E-Mails verwendet wird
<code>Host</code>	Eigenschaft, die die Host-Adresse des SMTP-Servers enthält
<code>SMTPAuth</code>	Eigenschaft zur Aktivierung der SMTP-Authentifizierung; auf <code>true</code> gesetzt, da ein Benutzername und ein Kennwort für die Anmeldung beim SMTP-Server erforderlich sind
<code>Username</code>	Eigenschaft für den Benutzernamen des SMTP-Kontos
<code>Password</code>	Eigenschaft für das Passwort des SMTP-Kontos
<code>Port</code>	Eigenschaft zur Angabe der Portnummer, die der SMTP-Server verwendet
<code>SMTPSecure</code>	Eigenschaft, die die zu verwendende Verschlüsselung festlegt; normalerweise auf <code>tls</code> gesetzt
<code>SMTPDebug</code>	Eigenschaft, die <code>PHPMailer</code> mitteilt, ob Debug-Informationen angezeigt werden sollen oder nicht
<code>isHTML()</code>	Methode, die <code>PHPMailer</code> mitteilt, dass die E-Mail HTML enthalten kann
<code>CharSet</code>	Eigenschaft für die in der E-Mail verwendete Zeichenkodierung; wenn diese nicht korrekt eingestellt ist, wird der Text im E-Mail-Programm des Empfängers möglicherweise nicht richtig dargestellt

Nachdem ein PHPMailer-Objekt erstellt und die Einstellungen konfiguriert wurden, kann eine E-Mail erstellt und versendet werden. Um eine E-Mail zu erstellen und an den SMTP-Server zu übergeben, müssen drei Methoden aufgerufen und drei Eigenschaften des PHPMailer-Objekts eingestellt werden.

Die Eigenschaften und Methoden in den ersten fünf Zeilen der folgenden Tabelle entsprechen dem Verfassen einer neuen E-Mail in einem E-Mail-Programm. Die verwendeten Werte können sich jedes Mal ändern, wenn die Website eine E-Mail sendet. Die letzte Methode entspricht dem Anklicken der send-Schaltfläche zum Verschicken der E-Mail.

EIGENSCHAFT/METHODE	BESCHREIBUNG
setFrom()	Methode zum Festlegen der Adresse, von der die E-Mail gesendet wird
addAddress()	Methode zum Festlegen der Adresse, an die die E-Mail gesendet werden soll (diese Methode kann erneut aufgerufen werden, um weitere Adressen hinzuzufügen)
Subject	Eigenschaft für die der Betreffzeile der E-Mail
Body	Eigenschaft, die den Textkörper der E-Mail enthält, die HTML verwendet
AltBody	Eigenschaft zur Aufnahme einer reinen Textversion der E-Mail (ohne HTML-Markup)
send()	Methode zur Verbindung mit dem SMTP-Server und zur Übermittlung der E-Mail

Um ein PHPMailer-Objekt zu erstellen, ihm mitzuteilen, wie es sich mit einem SMTP-Server verbinden soll, und dann die E-Mail zu erstellen und zu versenden, sind mindestens 18 Codezeilen erforderlich.

Da innerhalb einer Website oft mehrere Seiten die eine Transaktions-E-Mail senden müssen, kann die Website den gesamten Code in einer neuen benutzerdefinierten Klasse Email speichern, die Sie auf der nächsten Seite sehen, statt diesen Code auf jeder dieser Seiten zu wiederholen.

Jede Seite, die eine E-Mail senden muss, kann dies mit den beiden folgenden Anweisungen tun.

Die erste Zeile erstellt ein Objekt mit der benutzerdefinierten Email-Klasse und speichert es in der Variablen \$email. Die Konfigurationsdaten, die das Objekt benötigt, werden an die Konstruktormethode übergeben.

Die zweite Zeile ruft die Methode sendEmail() des Email-Objekts auf, um eine E-Mail zu erstellen und zu versenden.

```
① $email = new Email($email_config);
② $email->sendEmail($from, $to, $subject, $message);
```

Die neue benutzerdefinierte E-Mail-Klasse wird auf der nächsten Seite gezeigt; ihre beiden Methoden werden im Folgenden beschrieben.

Auf den folgenden zwei Seiten erhalten Sie ein Beispiel für die Verwendung der Klasse.

METHODE	BESCHREIBUNG
__construct(\$email_config)	Ein PHPMailer-Objekt erstellen und in der PHPMailer-Eigenschaft speichern. Konfigurieren, wie PHPMailer sich mit dem SMTP-Server verbinden soll. Diese Anweisungen gehören in die __construct()-Methode, da sie jedes Mal, wenn eine PHP-Seite eine E-Mail senden muss, gleich sind.
sendEmail(\$from, \$to, \$subject, \$message)	Eine E-Mail erstellen und an den SMTP-Server übergeben. Diese Methode wird aufgerufen, um eine E-Mail zu versenden. Bei jedem Aufruf können die Argumente unterschiedliche Werte haben.

KLASSEN ZUM ERSTELLEN UND SENDEN VON E-MAILS

Die Klasse `Email` wird verwendet, wenn eine Seite eine E-Mail versenden muss. Sie enthält den Code, mit dem ein `PHPMailer`-Objekt erstellt, eine E-Mail generiert und zum Versand an einen SMTP-Server übergeben wird.

1. Die Klasse erhält den Namensraum `PhpBook\Email`.

2. Die Eigenschaft `$phpmailer` speichert ein `PHPMailer`-Objekt. Sie wird als geschützte Eigenschaft deklariert, damit sie nur vom Code dieser Klasse verwendet werden kann.

3. Die Methode `__construct()` hat einen Parameter: die in `$email_config` gespeicherten Konfigurationsdaten.

Die Aufgaben innerhalb dieser Methode müssen jedes Mal ausgeführt werden, wenn eine Seite eine E-Mail senden muss:

- Sie erstellen ein `PHPMailer`-Objekt.
- Sie konfigurieren die Verbindung mit dem SMTP-Server.
- Sie stellen die Zeichenkodierung und die Art der E-Mail ein.

4. Das `PHPMailer`-Objekt wird erstellt und in der Eigenschaft `$phpmailer` des `Email`-Objekts gespeichert. (Das Argument `true` weist `PHPMailer` an, eine Ausnahme zu werfen, wenn ein Problem beim Erstellen oder Versenden einer E-Mail auftritt.)

5. Die Methode `isSMTP()` des `PHPMailer`-Objekts zeigt an, dass die E-Mail über einen SMTP-Server gesendet wird.

6. Die `SMTPAuth`-Eigenschaft wird auf `true` gesetzt, weil ein Benutzername und ein Passwort für die Anmeldung am SMTP-Server erforderlich sind.

7. Die für die Verbindung mit dem SMTP-Server erforderlichen Daten werden mit den Werten aus dem Array `$email_config` (das bei der Erstellung des Objekts an die Konstruktormethode übergeben wurde) festgelegt.

8. `PHPMailer` wird mitgeteilt, dass die Zeichenkodierung `UTF-8` ist und dass HTML-E-Mails versandt werden.

9. Die Methode `sendEmail()` erstellt und versendet einzelne E-Mails. Sie hat vier Parameter für die Daten, die sich jedes Mal ändern können, wenn das Objekt zum Senden einer E-Mail verwendet wird.

- `$from` gibt an, von wem die E-Mail gesendet wird.
- `$to` enthält die E-Mail-Adresse des Empfängers.
- `$subject` enthält die Betreffzeile der E-Mail.
- `$message` enthält die zu versendende Nachricht.

Wenn die E-Mail erstellt und versendet wurde, wird `true` zurückgegeben.

10. Die `setFrom()`-Methode legt die E-Mail-Adresse fest, von der die E-Mail gesendet werden soll.

11. Die `addAddress()`-Methode legt die E-Mail-Adresse fest, an die die Nachricht gesendet werden soll.

12. Die Eigenschaft `Subject` legt den Betreff der E-Mail fest.

13. Die `Body`-Eigenschaft legt den Textkörper der E-Mail fest. Sie besteht aus einigen grundlegenden HTML-Tags für den Start der HTML-E-Mail, gefolgt von dem Wert, der im `$message`-Parameter stand, und dann den abschließenden HTML-Tags.

14. Die Eigenschaft `AltBody` legt die reine Textversion der E-Mail fest. Sie verwendet die PHP-Funktion `strip_tags()`, um Markup aus der Nachricht zu entfernen (siehe Hinweis rechts).

15. `send()` übergibt die E-Mail an den SMTP-Server.

16. Die Methode gibt `true` zurück, um anzugeben, dass die E-Mail erstellt und an den SMTP-Server weitergeleitet wurde.

(Sie hätte eine Ausnahme geworfen, wenn es nicht funktioniert hätte.)

Als Nächstes werden Sie sehen, wie Sie die Klasse zum Versenden einer E-Mail verwenden können. Sobald eine Seite ein Objekt mit der Klasse `Email` erstellt und eine E-Mail gesendet hat, kann sie weitere E-Mails senden, indem sie die Methode `sendEmail()` erneut aufruft.

```

<?php
① Namensraum PhpBook\Email; // Namensraum deklarieren

class Email {

② protected $phpmailer; // PHPMailer-Objekt

③ public function __construct($email_config)
{
    $this->phpmailer = new \PHPMailer\PHPMailer\PHPMailer(true); // PHPMailer erstellen
    $this->phpmailer->isSMTP(); // SMTP verwenden
    $this->phpmailer->SMTPAuth = true; // Authentifizierung ein
    $this->phpmailer->Host = $email_config['server']; // Serveradresse
    $this->phpmailer->SMTPSecure = $email_config['security']; // Security-Art
    $this->phpmailer->Port = $email_config['port']; // Port
    $this->phpmailer->Username = $email_config['username']; // Nutzernname
    $this->phpmailer->Password = $email_config['password']; // Kennwort
    $this->phpmailer->SMTPDebug = $email_config['debug']; // Debug-Methode
    $this->phpmailer->CharSet = 'UTF-8'; // Zeichenkodierung
    $this->phpmailer->isHTML(true); // HTML-E-Mail
}

⑨ public function sendEmail($from, $to, $subject, $message): bool
{
    $this->phpmailer->setFrom($from); // Von E-Mail-Adresse
    $this->phpmailer->addAddress($to); // An E-Mail-Adresse
    $this->phpmailer->Subject = $subject; // Betreff der E-Mail
    $this->phpmailer->Body = '<!DOCTYPE html><html lang="en-us"><body>' . $message . '</body></html>'; // E-Mail-Text
    $this->phpmailer->AltBody = strip_tags($message); // Textkörper
    $this->phpmailer->send(); // E-Mail senden
    return true; // true zurückgeben
}
}

```

Wenn eine HTML-E-Mail versendet wird, wird zusammen mit der HTML-Version eine Nur-Text-Version der E-Mail versendet. Es ist wichtig, diese Version der E-Mail zu erstellen, da Spam-Filter gerne eine Nur-Text-Version der E-Mail sehen (und manche Leute verwenden Nur-Text-E-Mail-Programme).

Die in PHP eingebaute Funktion `strip_tags()` soll Tags aus dem Markup entfernen. Ihr Parameter ist eine Zeichenkette mit Auszeichnungselementen, und sie gibt die Zeichenkette mit den entfernten Tags zurück. Sie können auch HTML Purifier verwenden, um Markup aus Ihren E-Mails zu entfernen.

DIE E-MAIL-KLASSE VERWENDEN

Eine neue Seite contact.php (siehe rechte Seite) enthält ein Formular zum Versenden einer E-Mail an die Betreiber der Website. Um die E-Mail zu senden, erstellt die Kontaktseite ein Objekt mit der Klasse Email und ruft dann die Methode sendEmail() mit vier Parametern auf: die E-Mail, von der sie stammt, die, an die sie gesendet werden soll, die Betreffzeile und die Nachricht.

1. Der use-Befehl importiert den Code aus der Klasse Validate in den aktuellen Namensraum.
2. Wenn das Formular abgeschickt wurde, werden die E-Mail-Adresse und die Nachricht abgerufen und in Variablen gespeichert.
3. Die eingegebenen Werte werden mithilfe der Klasse Validate überprüft. Alle Fehlermeldungen im Array werden zusammengeführt und in der Variablen \$invalid gespeichert.
4. Wenn die Daten ungültig waren, enthält \$errors eine Meldung.

5. Andernfalls versucht die Seite, die E-Mail zu senden.
6. Der Betreff der E-Mail wird in \$subject gespeichert.
7. Ein Email-Objekt wird mithilfe der Email-Klasse erstellt.
8. Die Methode sendEmail() des Email-Objekts wird aufgerufen, um die E-Mail zu erstellen und zu versenden. Sie hat vier Argumente:
 - die Adresse, von der die E-Mail gesendet wird
 - die Adresse, an die die E-Mail gesendet werden soll
 - die Betreffzeile
 - die Nachricht
9. Wenn keine Ausnahme ausgelöst wurde, enthält die Variable \$success eine Erfolgsmeldung für den Benutzer.
10. Das Array \$data enthält die Daten für die Seite, und die Twig-Methode render() generiert den HTML-Code.

c15/templates/contact.html

Twig

```
{% extends 'layout.html' %}  
{% block content %}  
<main class="container" id="content">  
  <section class="heading"><h1>Contact Us</h1></section>  
  <form method="post" action="contact.php" class="form-contact">  
    {% if errors.warning %}<div class="alert-danger">{{ errors.warning }}</div>{% endif %}  
    {% if success %}<div class="alert-success">{{ success }}</div>{% endif %}  
    <label for="email">Email: </label>  
    <input type="text" name="email" id="email" value="{{ from }}" class="form-control">  
    <span class="errors">{{ errors.email }}</span><br>  
    <label for="message">Message: </label><br>  
    <textarea id="message" name="message" class="form-control">{{ message }}</textarea>  
    <span class="errors">{{ errors.message }}</span><br>  
    <input type="submit" value="Submit Message" class="btn">  
  </form>  
</main>  
{% endblock %}
```

```

<?php
declare(strict_types = 1);
① use PhpBook\Validate\Validate;
include '../src/bootstrap.php';
$from    = '';
$message = '';
$errors  = [];
$success = '';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {      // Wenn Formular abgeschickt
②   $from        = $_POST['email'];                // E-Mail Adresse
   $message     = $_POST['message'];              // Meldung
   $errors['email'] = Validate::IsEmail($from)      ? '' : 'Email not valid';
   $errors['message'] = Validate::IsText($message, 1, 1000) ? '' : 'Please enter a
③       message up to 1000 characters';
   $invalid = implode($errors);                  // Alle Fehlermeldungen
   if ($invalid) {
      $errors['warning'] = 'Please correct the errors'; // Warnung
④   } else {
      $subject = "Contact form message from " . $from; // Nachrichtentext
      $email   = new \PhpBook\Email\Email($email_config); // E-Mail-Objekt
      $email->sendEmail($email_config['admin_email'], $email_config['admin_email']),
      $subject, $message);                         // Senden
      $success = 'Your message has been sent';      // Erfolgsmeldung
⑤   }
}
⑥   $data['navigation'] = $cms->getCategory()->getAll(); // Alle Navigationskategorien
⑦   // Die folgenden Werte werden nur erstellt, wenn das Formular abgesandt wurde
⑧   $data['from']      = $from;                          // von E-Mail
⑨   $data['message']   = $message;                     // Meldung
⑩   $data['errors']    = $errors;                       // Fehlermeldungen
   $data['success']   = $success;                      // Erfolgsmeldung
echo $twig->render('contact.html', $data);           // Vorlage darstellen

```

ERGEBNIS

CONTACT US

Email:

Message:

SUBMIT MESSAGE

HINWEIS: Wenn die DEV-Konstante in config.php auf true gesetzt ist (siehe Schritt 7 auf S. 595), erstellt PHPMailer eine lange Liste von Debug-Meldungen, die auf der Seite vor der Kopfzeile und dem Formular angezeigt werden.

Ist die DEV-Konstante auf false, gesetzt, werden die Meldungen nicht angezeigt.

ZUSAMMENFASSUNG

NAMENSRÄUME & BIBLIOTHEKEN

- **Namensräume** stellen sicher, dass der PHP-Interpreter zwischen zwei oder mehreren Klassen, Funktionen oder Konstanten unterscheiden kann, wenn diese den gleichen Namen haben.
- **Bibliotheken und Pakete** ermöglichen es Ihnen, von anderen geschriebenen Code zu verwenden.
- **Composer** hilft bei der Verwaltung der Pakete, die eine Website verwendet.
- **Packagist.org** listet Pakete auf, die Composer verwenden kann.
- **Composer** erstellt einen Autoloader, der Klassendateien für die Pakete enthält, wenn eine Seite diese verwendet.
- Mit **HTML Purifier** kann Markup entfernt werden, das einen XSS-Angriff verursachen könnte.
- **Twig** ist eine Template-Engine, die den PHP-Code, der Daten abruft und verarbeitet, von Templates trennt, die die Darstellung dieser Daten steuern.
- **PHPMailer** ist ein Paket, mit dem E-Mails mit PHP erstellt und an einen SMTP-Server weitergeleitet werden können.

16

MITGLIEDSCHAFT

In diesem Kapitel erfahren Sie, wie sich Besucher als Mitglieder einer Website registrieren können. Anschließend können sie sich anmelden und Seiten sehen, die nur für Mitglieder angezeigt werden und für sie personalisiert sind.

Wenn sich ein Mitglied auf einer Website registriert, muss es in der Regel folgende Angaben machen:

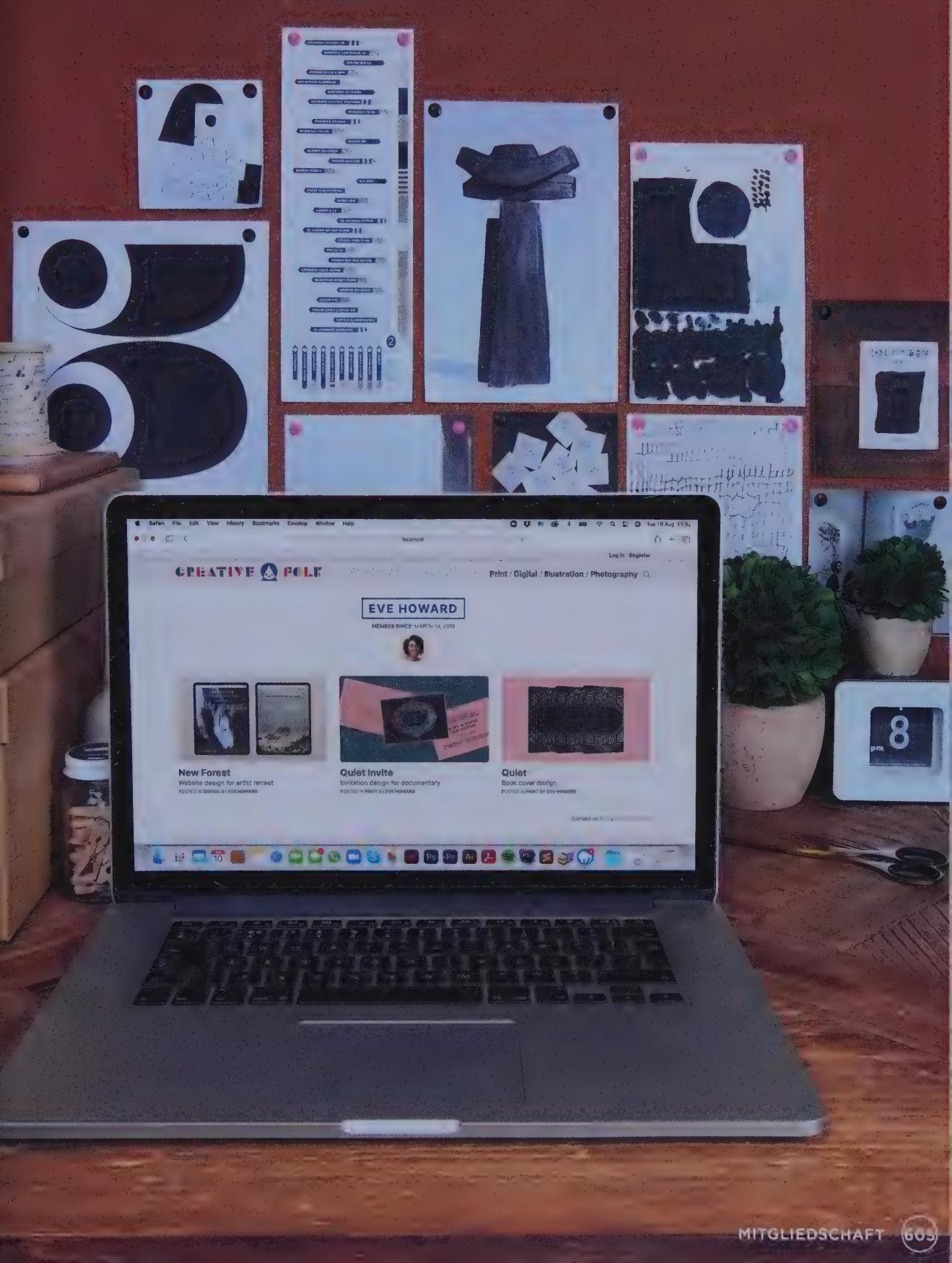
- eine **Kennung**: Sie dient zur Identifizierung des Mitglieds, z.B. eine E-Mail-Adresse oder ein Benutzername. Jedes Mitglied der Website benötigt seine eigene eindeutige Kennung.
- ein **Passwort**, mit dem bestätigt wird, dass das Mitglied die Person ist, als die es sich ausgibt. Der Nutzer ist die einzige Person, die das Passwort kennen sollte.

Diese Daten werden in der Datenbank gespeichert. Wenn sich ein Mitglied auf der Beispiel-Website angemeldet hat, kann es:

- Seiten ansehen, die nur für Mitglieder zugänglich sind
- sein eigenes Profil erstellen und bearbeiten
- seine eigenen Arbeiten hochladen

Dieses Kapitel ist in drei Abschnitte unterteilt:

- **Registrierung für eine Website**: Sie lernen, wie man die Informationen sammelt, die zur Identifizierung eines Mitglieds der Website erforderlich sind, und wie man sie in der Datenbank speichert.
- **Einloggen und Seiten personalisieren**: Wie man es Mitgliedern ermöglicht, sich anzumelden, Seiten erstellt, die auf einzelne Mitglieder zugeschnitten sind, und wie man Seiten nur für Mitglieder erstellt.
- **Aktualisierung der Datenbank, ohne dass der Benutzer sich anmelden muss**: Wie man Benutzer die Datenbank aktualisieren lässt, ohne dass sie sich vorher angemeldet haben, etwa wenn sie ein Passwort aktualisieren müssen. Dies erfordert eine Reihe neuer Sicherheitsanforderungen.



DATENBANK AKTUALISIEREN

Für die letzten beiden Kapitel benötigt die Datenbank der Beispiel-Website drei zusätzliche Tabellen und neue Spalten in mehreren bestehenden Tabellen.

Gemäß den auf S. 392 beschriebenen Schritten verwenden Sie PHPMyAdmin, um:

- die neue Datenbank `phpbook-2` zu erstellen
- die Datei `phpbook-2.sql` (im Code-Download zu diesem Kapitel) zu importieren, um die Tabellen in der neuen Datenbank zu erstellen und ihnen Daten hinzuzufügen

Nachdem Sie diese neue Datenbank erstellt haben, sehen Sie sich die Änderungen in PHPMyAdmin an. Zunächst gibt es drei neue Tabellen. Die `token`-Tabelle wird am Ende dieses Kapitels vorgestellt. Die Tabellen `comment` und `likes` werden im nächsten Kapitel vorgestellt.

Außerdem gibt es neue Spalten in den `article`-, `category`- und `member`-Tabellen. Die neue Spalte `role` in der Tabelle `member` wird in diesem Kapitel erläutert. Die neue Spalte `seo_title` in der Tabelle `article` und die Spalte `seo_name` in der Tabelle `category` werden im nächsten Kapitel vorgestellt.

Nachdem Sie sich die Änderungen an der Datenbank in PHPMyAdmin angesehen haben, öffnen Sie die Datei `config.php` im Code dieses Kapitels und fügen die Einstellungen für die Verbindung zur neuen Datenbank hinzu. Die einzige Einstellung, die sich von den vorherigen Kapiteln unterscheidet, ist der neue Datenbankname (`phpbook-2`).

Sobald Sie den Beispielcode für dieses Kapitel auf Ihrem Computer ausgeführt haben, verwenden Sie den Registrierungslink in der rechten oberen Ecke jeder Seite, um Ihr eigenes Konto zu erstellen. Sobald Sie sich registriert haben, loggen Sie sich ein und sehen sich in dieser Version der Website um.

Das sollten Sie sehen:

- Es gibt mehr Mitglieder (und deren Arbeit).
- Sie können nur auf die Admin-Seiten zugreifen, wenn Sie eingeloggt sind (und diese sehen dürfen).

Am Ende dieses Kapitels werden Sie gelernt haben, wie der Code es den Benutzern ermöglicht, sich zu registrieren, anzumelden, ihre eigenen Arbeiten hochzuladen und einen Link zum Zurücksetzen des Passworts anzufordern.

HINWEIS: Der Code, der für dieses Kapitel heruntergeladen wurde, enthält einige Dateien, die in diesem Buch nicht beschrieben werden, da die von Ihnen erfüllten Aufgaben bereits in anderen Beispielen beschrieben wurden.

Die Seite, auf der Mitglieder ihre Arbeiten hochladen können, entspricht beispielsweise der Seite, auf der Administratoren Artikel erstellen können; der Hauptunterschied besteht darin, dass die Mitglieder ein Bild hochladen müssen und ihre Artikel automatisch veröffentlicht werden. Auch die Seite, auf der die Mitglieder ihre Profile bearbeiten können, funktioniert wie die Seite, auf der Administratoren Kategorien bearbeiten können.

BENUTZER REGISTRIEREN

Die Besucher müssen das Registrierungsformular ausfüllen, um Mitglied der Website zu werden, und ihre Angaben werden in der Tabelle member der Datenbank gespeichert.

The screenshot shows a registration form with the following fields:

- Forename:
- Surname:
- Email address:
- Password:
- Confirm Password:

Below the fields is a blue button labeled "REGISTER".

Die Seite register.php verwendet das oben gezeigte Formular, um Besuchern die Möglichkeit zu geben, sich als Mitglied der Website zu registrieren.

Nach dem Absenden des Formulars werden die Daten überprüft, und eine neue Methode der Klasse Member namens `create()` fügt die Daten in die member-Tabelle ein, wobei die in Kapitel 14 erläuterten Techniken verwendet werden.

Die Registrierung nutzt zwei neue Konzepte:

- **Rollen** steuern, welche Aufgaben ein Benutzer der Website ausführen darf.
- **Passwort-Hashes** werden von Websites anstelle des eigentlichen Passworts gespeichert, das das Mitglied bei der Registrierung eingegeben hat.

ROLLEN

Websites erlauben oft, dass verschiedene Mitglieder verschiedene Seiten ansehen und verschiedene Aufgaben ausführen können. Über eine Rolle wird definiert, welche Aufgaben ein Mitglied ausführen kann. Auf der Beispelseite wird unterschieden zwischen:

- **einem Besucher:** eine Person, die nicht eingeloggt ist. Sie kann nur die Arbeit auf der Website ansehen.
- **einem Mitglied:** jemand, der sich registriert und eingeloggt hat. Er kann sein Mitgliedsprofil bearbeiten, neue Werke hochladen und bestehende Werke bearbeiten.
- **einem gesperrten Mitglied:** jemand, der sich registriert hat, der die Website jedoch nicht mehr nutzen kann, weil er gesperrt wurde und sich nicht mehr anmelden kann.
- **einem Administrator:** der Betreiber der Website oder eine Person, die an der Website arbeitet. Sie kann Admin-Seiten aufrufen, Kategorien erstellen, Arbeiten löschen und die Rollen der Benutzer aktualisieren.

In der member-Tabelle wird in der neuen Spalte `role` die Rolle jedes Mitglieds gespeichert. Ihr Wert ist `member`, `suspended` oder `admin` (`viristor` wird nicht gespeichert, da dies für Personen gilt, die sich nicht angemeldet haben).

HINWEIS: Wenn sich Benutzer für die Beispieleiste registrieren, wird ihre Rolle auf `admin` gesetzt, da dies Ihnen den Zugriff auf die Admin-Seiten ermöglicht, ohne dass die Rolle in der Datenbank manuell geändert werden muss. Auf einer Live-Site wäre die Standardrolle für neue Mitglieder `member`, und ein Administrator könnte diese Rolle ändern.

PASSWORT-HASHES

Aus Sicherheitsgründen sollten Websites die Passwörter ihrer Mitglieder nicht speichern, sondern stattdessen eine verschlüsselte Version des Passworts, einen sogenannten **Hash**. Es ist nicht möglich, einen Hash wieder in den ursprünglichen Text zu entschlüsseln.

Nur ein Mitglied sollte sein eigenes Passwort kennen. Eine Website kann das Passwort eines Mitglieds überprüfen, aber die Passwörter sollten nicht in der Datenbank gespeichert werden.

Wenn Mitglieder sich anmelden, verschlüsselt ein Algorithmus (eine Reihe von Regeln) ihre Passwörter in einen Hash, der wie eine zufällige Menge alphanumerischer Zeichen aussieht. Die Datenbank speichert den Hash anstelle des Kennworts.

EINGABE: PASSWORT

Wenn ein Mitglied ein Passwort eingibt, konvertiert eine Funktion es in einen Hash:

ivy@eg.link
BlueRothkoBath23!



Ein Hash kann nicht in das ursprüngliche Kennwort zurückverwandelt werden. Selbst wenn jemand Zugang zur Datenbank hätte, könnte er nicht an die Kennwörter der Mitglieder gelangen.

Dies schützt sowohl Ihre eigene Website als auch Mitglieder, die das gleiche Passwort auf anderen Websites verwendet haben.

Unabhängig von der Anzahl der Zeichen des Kennworts ist der Hash immer gleich lang (in diesem Buch sind es 60 Zeichen), sodass der Hash keine Rückschlüsse auf die Länge des Kennworts zulässt.

PHP verfügt über integrierte Funktionen zur Erstellung von Hashes.

Jedes Mal, wenn die Funktion verwendet wird, um das Passwort in einen Hash umzuwandeln, wird derselbe Satz von Zeichen erzeugt.

Wenn sich ein registriertes Mitglied auf der Website anmeldet, wird das eingegebene Passwort erneut durch den Hash-Algorithmus geleitet. Wenn dieser Wert mit dem in der Datenbank gespeicherten Hash übereinstimmt, hat der Benutzer das richtige Passwort eingegeben.

ERGEBNIS: HASH

Die Datenbank speichert den Hash (und nicht das tatsächliche Passwort):

email	password
ivy@eg.link	\$2y\$10\$XTeGk6Z7XG1Gs
BlueRothkoBath23!	26.MVvCIOANsdgFjZOYE MDWY1mlca4cOKyMwjufi

Um den Hash noch sicherer zu machen, fügt PHP dem Passwort eine zufällige Buchstabenfolge hinzu, die **Salt** genannt wird. Meldet sich der Benutzer erneut an,

- ermittelt PHP das Salt aus dem gespeicherten Hash des Benutzers,
- wird ein Hash des Passworts erstellt, das der Benutzer beim Einloggen verwendet hat,
- wird das Salt zum neuen Passwort-Hash hinzugefügt,
- wird der gespeicherte mit dem neuen Wert verglichen.

Wenn beide übereinstimmen, ist das Passwort korrekt.

GEHASHTE KENNWÖRTER ERSTELLEN UND PRÜFEN

Die Funktion `password_hash()` von PHP erzeugt einen Hash aus einem Passwort. Die PHP-Funktion `password_verify()` prüft, ob das vom Besucher angegebene Passwort denselben Hash erzeugt, der bereits gespeichert wurde.

Die in PHP eingebaute Funktion `password_hash()` nimmt ein Passwort und gibt einen Hash zurück. Sie hat drei Parameter:

- das Passwort, das gehasht werden soll
- der Name des zu verwendenden Hash-Algorithmus
- ein optionales Array mit Einstellungen für diesen Algorithmus (auf der Beispelseite sind keine Optionen gesetzt)

Die PHP .net-Site gibt eine Reihe von Konstanten an, die verwendet werden können, um den Namen des vom Code verwendeten Hash-Algorithmus anzugeben: http://notes.re/php/pwd_hash/. Die Beispelseite verwendet den Namen `PASSWORD_DEFAULT` für den Standard-Hashing-Algorithmus von PHP. Im Moment ist dies der `bcrypt`-Algorithmus, aber das kann sich mit der Entwicklung stärkerer Algorithmen ändern.

```
password_hash($password, $algorithm[, $options]);
```

PASSWORT

ALGORITHMUS

OPTIONEN

Die PHP-Funktion `password_verify()` nimmt ein vom Benutzer eingegebenes Passwort und erstellt einen Hash.

Dann vergleicht sie diesen Wert mit dem Hash, der bereits für ihn gespeichert wurde. Wenn beide Werte übereinstimmen, hat der Benutzer das richtige Passwort angegeben. Die Funktion `password_verify()` hat zwei Parameter:

- das Passwort, das das Mitglied gerade angegeben hat
- den Hash-Wert, der bereits für dieses Mitglied gespeichert wurde

Sie müssen den Algorithmus oder das Salt, das bei der Erstellung des Hashes verwendet wurde, nicht angeben, da die Funktion `password_verify()` diese Einstellungen anhand des gespeicherten Hashes ermitteln kann.

Die Funktion gibt zurück:

- `true`, wenn die Hashes übereinstimmen
- `false`, wenn es sich um unterschiedliche Werte handelt

```
password_verify($password, $hash);
```

EINGEGEBENES
PASSWORD

IN DATENBANK GESPEICHERTER
HASH

REGISTRIERUNG NEUER MITGLIEDER (TEIL 1)

Die Registrierungsseite funktioniert wie die Admin-Seiten, mit denen Daten in die Datenbank eingegeben werden. Nach dem Abschicken des Formulars werden die Daten überprüft. Sind sie gültig, fügt eine neue Methode der Klasse Member das Mitglied zur Datenbank hinzu.

Die Datei `register.php` ermöglicht es Besuchern, sich als Mitglied der Website zu registrieren (auf S. 607 wurde gezeigt, wie diese Seite im Browser aussieht). Wenn Besucher das Formular ausfüllen und gültige Daten angeben, werden sie mit der `create()`-Methode des Member-Objekts (die Sie auf S. 612 kennenlernen werden) zur Datenbank hinzugefügt. Wenn die Daten ungültig sind, werden stattdessen Fehlermeldungen angezeigt.

1. Die strikte Typisierung ist aktiviert und der `validate`-Namensraum importiert, sodass er in der Seite verwendet werden kann, ohne dass der vollständige Namespace eingegeben werden muss.
2. Die Datei `bootstrap.php` wird in die Seite eingebunden.
3. Die Arrays `$member` und `$errors` werden als leere Arrays initialisiert, damit sie dem von den Twig-Templates (in Schritt 12) verwendeten Array `$data` hinzugefügt werden können, auch wenn ihnen in den Schritten 4–11 keine Daten hinzugefügt werden.
4. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
5. Wenn dies der Fall ist, werden die Daten aus dem Formular abgerufen. Der Wert aus dem Kennwortbestätigungsfeld wird in einer separaten Variablen gespeichert, da sein Wert nicht zur Datenbank hinzugefügt wird (mit ihm wird nur bestätigt, dass der Benutzer dasselbe Kennwort zweimal eingegeben hat).
6. Die Daten werden validiert. Wenn Daten ungültig sind, werden Fehlermeldungen im Array `$errors` gespeichert. Es gibt zwei neue Methoden in der `Validate`-Klasse für dieses Kapitel; sie prüfen, ob die E-Mail-Adresse gültig ist und das Passwort den Mindestanforderungen entspricht.

7. Alle Werte im Array `$errors` werden zum String `$invalid` zusammengefasst.
8. Eine `if`-Anweisung prüft, ob `$invalid` keinen Text enthält. Ist dies nicht der Fall, sind die Daten gültig. Wenn ja, enthält das `$errors`-Array mindestens eine Fehlermeldung.
9. Wenn die Daten gültig sind, wird die `create()`-Methode des Member-Objekts aufgerufen, um das Mitglied zur Datenbank hinzuzufügen (diese Methode wird auf S. 612 vorgestellt).
10. Die `create()`-Methode gibt `true` zurück, wenn das Mitglied erfolgreich hinzugefügt wurde, oder `false`, wenn die E-Mail-Adresse bereits verwendet wird (wenn es ein anderes Problem gibt, wird eine Ausnahme ausgelöst). Der zurückgegebene Wert wird in `$result` gespeichert.
11. Eine `if`-Anweisung prüft, ob `$result` den Wert `false` enthält. Ist dies der Fall, speichert der Schlüssel `email` des Arrays `$errors` eine Nachricht, die dem Benutzer mitteilt, dass die E-Mail-Adresse bereits verwendet wird.
12. Andernfalls wurde der Benutzer in Schritt 9 zur Datenbank hinzugefügt, sodass er zur Anmeldeseite weitergeleitet und eine Erfolgsmeldung im Abfragestring gesendet wird.
13. Die Daten, die das Twig-Template anzeigen muss, werden im Array `$data` gespeichert.
14. Die `render()`-Methode des Twig-Umgebungsobjekts wird aufgerufen, um den HTML-Code zu erstellen, der an den Browser zurückgesendet wird. Sie verwendet das Template `register.html`.

```

<?php
① declare(strict_types = 1);
use PhpBook\Validate\Validate;

② include '../src/bootstrap.php';
③ $member = [];
④ $errors = [];

⑤ if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Get form data
    $member['forename'] = $_POST['forename'];
    $member['surname'] = $_POST['surname'];
    $member['email'] = $_POST['email'];
    $member['password'] = $_POST['password'];
    $confirm = $_POST['confirm'];

    // Formulardaten validieren
    $errors['forename'] = Validate::isText($member['forename'], 1, 254)
        ? '' : 'Forename must be 1-254 characters';
    $errors['surname'] = Validate::isText($member['surname'], 1, 254)
        ? '' : 'Surname must be 1-254 characters';
    $errors['email'] = Validate::isEmail($member['email'])
        ? '' : 'Please enter a valid email';
    $errors['password'] = Validate::isPassword($member['password'])
        ? '' : 'Passwords must be at least 8 characters and have:<br>
            A lowercase letter<br>An uppercase letter<br>A number
            <br>And a special character';
    $errors['confirm'] = ($member['password'] == $confirm)
        ? '' : 'Passwords do not match';
    $invalid = implode($errors);
}

⑥ if (!$invalid) {
    $result = $cms->getMember()->create($member);
    if ($result == false) {
        $errors['email'] = 'Email address already used'; // Warnung
    } else {
        redirect('login.php', ['success' => 'Thanks for joining! Please log in.']);
    }
}

⑦ $data['navigation'] = $cms->getCategory()->getAll();
⑧ $data['member'] = $member;
⑨ $data['errors'] = $errors;

⑩ echo $twig->render('register.html', $data);

```

// Strikte Typisierung
// Klasse Validate importieren

// Datei einrichten
// member-Array initialisieren
// Fehlerarray initialisieren

// Wenn Formular abgeschickt

// Vorname abfragen
// Nachname abfragen
// E-Mail abfragen
// Passwort abfragen
// Passwortbestätigung abrufen

// Fehlermeldungen

// Wenn keine Fehler
// Mitglied erstellen
// Wenn Ergebnis falsch
// Sonst zum Login

// Navigationskategorien
// Mitgliederdaten
// Fehlermeldungen

// Vorlage darstellen

REGISTRIERUNG NEUER MITGLIEDER (TEIL 2)

Auf der rechten Seite zeigt das obere Codefeld das Twig-Template, das für die Anzeige des Registrierungsformulars verwendet wird.

Das Formular im Code-Download enthält einige weitere HTML-Elemente und -Attribute, die zur Steuerung der Darstellung verwendet werden; einige von ihnen wurden aus diesem Code-Kasten entfernt, damit Sie sich auf seine Funktionalität konzentrieren können und damit es auf die Seite passt.

1. Das Template erweitert die Datei `layout.html` und stellt neue Inhalte für die Blöcke `title` und `description` bereit. (Diese Blöcke überschreiben den Standardtext, der in den `<title>`- und `<meta>`-Tags in dem auf S. 591 gezeigten Template `layout.html` enthalten war.)
2. In den Inhaltsblock wird das Registrierungsformular eingefügt.
3. Das Formular wird an dieselbe PHP-Seite übermittelt.
4. Waren die Formulardaten ungültig, wird dem Besucher eine Warnmeldung angezeigt.
5. Die Steuerelemente des Formulars werden angezeigt.

Das Formular kann auf zwei Arrays zugreifen, die Daten enthalten, falls das Formular abgeschickt wurde, die Daten aber nicht gültig waren:

- `member` ist ein Array mit den Daten, die der Benutzer eingegeben hat. Damit werden die Steuerelemente des Formulars gefüllt, sodass der Besucher nicht alle Daten erneut eingeben muss.
- `errors` ist ein Array mit Fehlermeldungen für jedes Formularsteuerelement, das die Validierung nicht bestanden hat. Diese Meldungen werden nach den Formularsteuerelementen angezeigt.

Beide Arrays wurden als leere Arrays am Anfang der Datei `register.php` in Schritt 3 auf der vorherigen Seite initialisiert.

Die Methode `create()` der Klasse `Member` fügt der Datenbank ein Mitglied hinzu. Sie folgt dem gleichen Ansatz wie die Methode zum Erstellen von Kategorien (auf S. 498 – 503).

Wenn ein Mitglied hinzugefügt wurde, gibt die Methode `true` zurück.

Wenn die E-Mail-Adresse bereits verwendet wird, gibt sie `false` zurück.

6. `create()` nimmt einen Parameter entgegen: ein Array mit den Mitgliederdaten. Die Funktion gibt einen booleschen Wert zurück.
7. Die Funktion `password_hash()` ersetzt das vom Benutzer angegebene Passwort durch einen Hash.
8. Ein `try`-Block enthält den Code zum Hinzufügen der Mitgliedsdaten in die Datenbank.
9. Die SQL `INSERT`-Anweisung fügt den Vornamen, den Nachnamen, die E-Mail und den Hash in die `Member`-Tabelle der Datenbank ein. (Die Datenbank erstellt die Werte für die Spalten `id`, `joined` und `role`.)
10. Die SQL-Anweisung wird ausgeführt.
11. Wenn der Code in der Methode noch läuft, wurde die SQL-Anweisung erfolgreich ausgeführt, und die Methode gibt `true` zurück.
12. Wenn PDO auf ein Problem gestoßen ist, wurde eine Ausnahme ausgelöst, und der `catch`-Block wird ausgeführt.
13. Lautet der Fehlercode 1062, bedeutet dies, dass die E-Mail-Adresse bereits in der Datenbank vorhanden ist und das Hinzufügen der Daten eine Eindeutigkeitsbedingung verletzen würde, sodass die Funktion `false` zurückgibt.
14. Andernfalls wird die Ausnahme mit dem Schlüsselwort `throw` erneut ausgelöst, sodass sie von der Standardfunktion zur Ausnahmebehandlung behandelt werden kann.

```

①  [{% extends 'layout.html' %}]
②  [{% block title %}Register{% endblock %}]
③  [{% block description %}Register for Creative Folk{% endblock %}]
④  [{% block content %}]
    <main class="container" id="content">
        <section class="header"><h1>Register</h1></section>
        <form method="post" action="register.php" class="form-membership">
            {%- if errors %}<div class="alert alert-danger">Please correct errors</div>{%- endif %}
            <label for="forename">Forename: </label>
            <input type="text" name="forename" value="{{ member.forename }}" id="forename">
            <div class="errors">{{ errors.forename }}</div>
            <label for="surname">Surname: </label>
            <input type="text" name="surname" value="{{ member.surname }}" id="surname">
            <div class="errors">{{ errors.surname }}</div>
            <label for="email">Email address: </label>
            <input type="email" name="email" value="{{ member.email }}" id="email">
            <div class="errors">{{ errors.email }}</div>
            <label for="password">Password: </label>
            <input type="password" name="password" id="password">
            <div class="errors">{{ errors.password }}</div>
            <label for="confirm">Confirm password: </label>
            <input type="password" name="confirm" id="confirm">
            <div class="errors">{{ errors.confirm }}</div>
            <input type="submit" class="btn btn-primary" value="Register">
        </form>
    </main>
    {% endblock %}

```

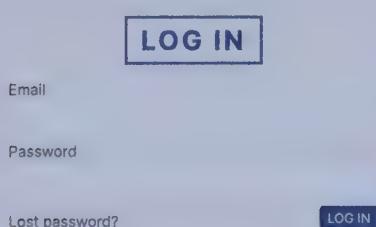
```

⑥  public function create(array $member): bool
    {
        $member['password'] = password_hash($member['password'], PASSWORD_DEFAULT); // Hash
        try {
            $sql = "INSERT INTO member (forename, surname, email, password)           // Versuch, Mitglied hinzuzufügen
                    VALUES (:forename, :surname, :email, :password)"; // Mitglied hinzuf.
            $this->db->runSQL($sql, $member);                      // SQL ausführen
            return true;
        } catch (\PDOException $e) {                                // Wenn PDOException
            if ($e->errorInfo[1] == 1062) {                         // Wenn Fehler doppelte Eintrag
                return false;                                       // false zurückgeben
            }
            throw $e;                                            // Sonst
                                                               // Ausnahme erneut auslösen
        }
    }

```

EINLOGGEN UND PERSONALISIEREN

Wenn Mitglieder auf die Website zurückkehren und sich anmelden, werden sie nach ihrer E-Mail-Adresse gefragt, um sich zu identifizieren, und nach ihrem Passwort, um zu bestätigen, dass sie die Person sind, für die sie sich ausgeben.



Auf der Seite `login.php` können sich die Mitglieder anmelden. Wird das Anmeldeformular abgeschickt, sucht eine neue Methode der Member-Klasse `Login()` in der Tabelle `member` der Datenbank nach der E-Mail-Adresse des Mitglieds und ruft dessen Daten ab, einschließlich des Passwort-Hashes.

Wenn das Passwort, das der Benutzer bei der Anmeldung angegeben hat, denselben Hash erzeugt, den die Datenbank für ihn gespeichert hat, geht die Website davon aus, dass das Mitglied die Person ist, für die sie sich ausgibt, und meldet es an.

Sobald sich ein Mitglied angemeldet hat, geschieht Folgendes:

- **Eine Session wird erstellt**, um Daten über das Mitglied und die Tatsache, dass es sich angemeldet hat, zu speichern.
- Die Seiten werden für dieses Mitglied mit Informationen **personalisiert**, die speziell für dieses Mitglied bestimmt sind.

SESSIONS

Sobald sich ein Mitglied eingeloggt hat, wird eine Session erstellt. Diese ermöglicht es der Website, das Mitglied jedes Mal zu identifizieren, wenn es während seines Besuchs auf der Website eine andere Seite aufruft. Sie speichert die ID, den Namen und die Rolle des Mitglieds, da diese in der Navigationsleiste verwendet werden, um ...

- einen Link zur Profilseite hinzuzufügen; der Link verwendet ihre Mitglieds-ID im Abfragestring,
- den Namen als Linktext für den Link anzuzeigen,
- einen Links zu den Admin-Seiten hinzuzufügen, wenn das Mitglied die `admin`-Rolle innehaltet.

Da jede Seite die Sessions benötigt, um die Navigationsleiste zu erstellen, wird eine neue Sessionklasse (S. 620–621) erstellt, um die Daten in der superglobalen `$_SESSION` zu nutzen:

- Ist der Benutzer eingeloggt, werden die Session-Daten des Benutzers zu den Eigenschaften des Session-Objekts hinzugefügt.
- Sonst werden die Werte dieser Eigenschaften automatisch mit Standardwerten belegt.

Die Klasse `Session` verfügt auch über Methoden zum Erstellen, Aktualisieren und Löschen von Sessions. Die Klasse fasst den für die Arbeit mit Sessions verwendeten Code zusammen und reduziert die Menge des auf jeder Seite benötigten Codes. Das Session-Objekt wird in `bootstrap.php` erstellt und in einer globalen Twig-Variablen zur Verfügung gestellt, damit alle Templates auf die Session-Daten zugreifen können.

Sobald sich ein Mitglied angemeldet hat, kann die Website Seiten erstellen, die auf der Grundlage der in der Datenbank gespeicherten Informationen auf diesen Nutzer zugeschnitten sind.

PERSONALISIERUNG

Sobald die Website ein einzelnes Mitglied identifizieren kann, kann sie die Seiten auf Grundlage der Präferenzen und des Profils dieses Nutzers anpassen.

Auf der linken Seite wurde bereits beschrieben, wie die Navigationsleiste bei der Anmeldung des Nutzers einen Link zu seiner Profilseite anzeigt und, wenn der Nutzer Administrator der Website ist, einen Link zu den Admin-Seiten.

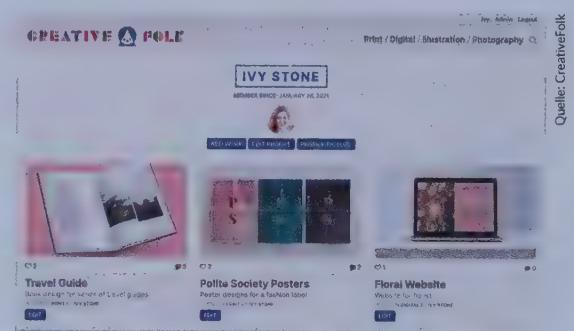
Wenn ein Benutzer seine Seite member.php aufruft, werden außerdem Links angezeigt, über die er seine Arbeit hinzufügen oder bearbeiten und sein Profil aktualisieren kann (wie rechts dargestellt).

Mit derselben Datei member.php werden die Details und Arbeiten jedes Mitglieds der Website angezeigt, aber diese zusätzlichen Links erscheinen nur, wenn ein Mitglied eingeloggt ist und seine eigene Seite betrachtet.

Neben der Erstellung personalisierter Seiten sichert dieser Abschnitt auch die Admin-Seiten, da sie nur von den Administratoren eingesehen werden sollten. Bis zu diesem Punkt konnte sie jeder einsehen.

Um die Admin-Seiten zu sichern, wird die neue Funktion `is_admin()` in die Datei functions.php eingefügt.

Sie wird am Anfang jeder Admin-Seite aufgerufen. Wenn der Benutzer nicht sowohl eingeloggt als auch Administrator ist, kann er die Seite nicht aufrufen.



Quelle: Creativefolk

Der Download-Code enthält die folgenden Dateien, die nicht abgedruckt sind, da sie bereits bekannten Dateien ähneln:

- `work.php` ermöglicht es Benutzern, ihre Arbeiten hochzuladen und zu bearbeiten. Sie entspricht `article.php` in Kapitel 14 (außer dass ein Bild erforderlich und `published` auf `true` gesetzt ist).
- `profile-edit.php` ermöglicht es Benutzern, ihr Profil zu bearbeiten. Sie entspricht der Admin-Seite, die zum Bearbeiten von Kategorien verwendet wird.
- `profile-pic-delete.php` löscht ein Profilbild. Sie entspricht der Datei `image-delete.php`, die zum Löschen von Artikelbildern verwendet wird.
- `profile-pic-upload.php` ermöglicht es Benutzern, ein neues Profilbild hochzuladen. Sie verwendet die gleiche Technik wie `article.php`, um Bilder hochzuladen.

LOGIN (TEIL 1)

Wenn ein Nutzer auf die Website zurückkehrt, kann er sich auf der Anmeldeseite anmelden. Wenn dieses Mitglied die richtigen Angaben macht, wird eine neue Session erstellt, in der Informationen während dieses Besuchs auf der Website gespeichert werden.

Die Seite `login.php` ermöglicht es den Mitgliedern, sich anzumelden.

1. Die strikte Typisierung ist aktiviert, der `Validate-Namensraum` importiert und `bootstrap.php` eingebunden.
2. Die Variablen `$email` und das Array `$errors` werden initialisiert. Sie werden benötigt, um das `$data`-Array zu erstellen, das die Twig-Templates in Schritt 15 benötigen.
3. Wenn der Abfragestring `success` enthält, wird der Wert in der Variablen `$success` gespeichert. (Diese wird dem Abfragestring hinzugefügt, wenn sich ein neuer Benutzer registriert.)
4. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
5. Wenn ja, werden die E-Mail-Adresse und das Passwort aus dem superglobalen Array `$_POST` entnommen und in den Variablen `$email` und `$password` gespeichert.
6. Die E-Mail-Adresse und das Passwort werden überprüft. Sind sie nicht gültig, werden Fehlermeldungen in den entsprechenden Schlüsseln des Arrays `$errors` gespeichert.
7. Die PHP-Funktion `implode()` fügt die Werte im Array `$errors` zu einem einzigen String zusammen, der in `$invalid` gespeichert wird.
8. Eine `if`-Anweisung testet, ob `$invalid` irgendwelche Fehlermeldungen enthält.
9. Falls ja, enthält der Schlüssel `message` des Arrays `$errors` eine Nachricht, die den Benutzer auffordert, es erneut zu versuchen.
10. Ansonsten waren die Anmeldedaten gültig ...

11. Die Methode `login()` der Klasse `Member` wird aufgerufen (siehe S. 618–619). Sie prüft, ob die E-Mail-Adresse in der Datenbank vorhanden ist und ob der Benutzer das richtige Passwort angegeben hat.

Wenn die Angaben korrekt sind, gibt die Methode die Daten des Mitglieds als Array zurück. Wenn nicht, gibt sie `false` zurück. Der zurückgegebene Wert wird in `$member` gespeichert.

Eine `if...elseif`-Anweisung verarbeitet die Ergebnisse:

12. Wenn die Variable `$member` die Daten eines Mitglieds enthält und dessen Rolle gesperrt ist, dann enthält das Array `$errors` eine Nachricht, die besagt, dass das Konto gesperrt ist.
13. Wenn `$member` einen Wert hat, hat sich das Mitglied erfolgreich angemeldet.
14. Eine Session wird für diesen Besucher mit der `create()`-Methode eines neuen Sessionsobjekts (siehe S. 620–621) erstellt.
15. Sobald der Benutzer eingeloggt ist, wird er zu seiner Profilseite weitergeleitet (und der Rest der Seite wird angehalten). Von diesem Punkt an ...
 - ersetzt die Navigation den Anmelde- durch einen Abmeldelink,
 - enthält die Navigation einen Link zur Profilseite des Mitglieds,
 - verlinkt die Navigation zum Adminbereich, wenn das Mitglied ein Administrator ist.
16. Andernfalls wurde kein Mitglied gefunden, sodass das Array `$errors` eine Nachricht enthält, die den Benutzer auffordert, es erneut zu versuchen.
17. Das Array `$data` enthält die Daten, die das Template benötigt, und die Twig-Methode `render()` erstellt die Seite.

```

<?php
declare(strict_types = 1);
use PhpBook\Validate\Validate;
① include '../src/bootstrap.php';

② $email   = '';
$errors  = [];
③ $success = $_GET['success'] ?? null;

④ if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email   = $_POST['email'];
    $password = $_POST['password'];
    $errors['email'] = Validate::isEmail($email)
        ? '' : 'Please enter a valid email address'; // E-Mail validieren
    $errors['password'] = Validate::isPassword($password)
        ? '' : 'Passwords must be at least 8 characters and have:<br>
            A lowercase letter<br>An uppercase letter<br>A number<br>
            And another character'; // Passwort validieren
    $invalid = implode($errors);
    // Fehler

    if ($invalid) {
        $errors['message'] = 'Please try again.'; // Wenn Daten ungültig
    } else {
        $member = $cms->getMember()->login($email, $password); // Mitgliederdetails
        if ($member and $member['role'] == 'suspended') { // Wenn Mitglied gesperrt
            $errors['message'] = 'Account suspended'; // Nachricht speichern
        } elseif ($member) { // Sonst für Mitglieder
            $cms->getSession()->create($member); // Session erstellen
            redirect('member.php', ['id' => $member['id'],]); // Auf ihre Seite leiten
        } else {
            $errors['message'] = 'Please try again.'; // Fehlermeldung speichern
        }
    }
}

$data['navigation'] = $cms->getCategory()->getAll(); // Navigationskategorien abrufen
$data['success']   = $success; // Erfolgsmeldung
⑭ $data['email']    = $email; // E-Mail-Adresse bei fehlgeschlagener Anmeldung
$data['errors']    = $errors; // Fehlerarray
echo $twig->render('login.html', $data); // Vorlage darstellen

```

Hinweis: Fehlermeldungen sollten nicht anzeigen, dass eine E-Mail-Adresse richtig ist, sondern dass ein Passwort falsch ist, da dies bestätigt, dass die E-Mail-Adresse für die Website registriert war.

Probieren Sie es: Wenn Sie das Session-Objekt (S. 620–621) verwendet haben, fügen Sie zwischen Schritt 2 und 3 eine if-Anweisung ein, um zu prüfen, ob sich das Mitglied bereits angemeldet hat. Wenn ja, leiten Sie das Mitglied auf seine member-Seite um.

LOGIN (TEIL 2)

Das erste Codefeld zeigt das Twig-Template für das Anmeldeformular. Der Code-Download enthält weitere HTML-Elemente und -Attribute zur Steuerung der Anzeige, aber einige davon wurden entfernt, damit Sie sich auf seine Funktionalität konzentrieren können und damit es auf die Seite passt.

1. Das Template erweitert `layout.html` und bietet neue Inhalte für die Blöcke `title` und `description`.
 2. Der Inhaltsblock enthält das Anmeldeformular.
 3. Das Formular wird an dieselbe PHP-Seite übermittelt.
 4. Wenn `success` einen Wert hat (also nicht `null` ist), hat sich ein neuer Benutzer registriert, und es wird eine Erfolgsmeldung angezeigt.
 5. Wenn das Array `errors` Werte enthält, wird der Wert für den Schlüssel `warning` angezeigt.
 6. Das Formular enthält Eingaben für E-Mail und Passwort. Wenn das Array `$errors` Fehlermeldungen enthält, werden diese nach dem entsprechenden Formularsteuerelement angezeigt (für die Passwort-Fehlermeldung wird der Twig-`raw`-Filter genommen, da er HTML-Markup verwendet – siehe Schritt 7 auf der vorherigen Seite).
- der zweite Codekasten zeigt eine neue `login()`-Methode in der `Member`-Klasse. Sie prüft, ob die E-Mail und das Passwort korrekt sind. Wenn ja, gibt sie die Details des Mitglieds zurück. Wenn nicht, gibt sie `false` zurück.
7. `login()` benötigt eine E-Mail-Adresse und ein Passwort.
 8. Die Variable `$sql` speichert eine SQL-Abfrage, um die Daten des Mitglieds unter Verwendung der E-Mail-Adresse abzurufen.
 9. Die SQL-Abfrage wird ausgeführt, und die Daten werden in `$member` gespeichert.
 10. Wenn die Daten des Mitglieds nicht gefunden wurden, gibt die Methode `login()` `false` zurück.

11. Wenn der Code in der Methode noch läuft, wurde das Mitglied gefunden. Als Nächstes erstellt die PHP-Funktion `password_verify()` einen Hash aus dem Passwort, das bei der Anmeldung angegeben wurde, und prüft, ob dieser mit dem Hash in der Datenbank übereinstimmt. Sie gibt `true` zurück, wenn sie übereinstimmen, `false`, wenn nicht. Das Ergebnis wird in der Variablen `$authenticated` gespeichert.

12. Ein ternärer Operator prüft, ob `$authenticated` den Wert `true` speichert. Wenn ja, gibt die Methode das Array `$member` zurück, sonst `false`.

Die letzte Codebox zeigt die Seite `bootstrap.php`, die den Code zum Einrichten der einzelnen Seiten enthält. Wie Sie gesehen haben, werden bei der Anmeldung eines Mitglieds dessen ID, Vorname, Nachname und Rolle in einer Session gespeichert, da jede Seite auf diese Session-Daten zugreifen muss, um die Navigationsleiste zu erstellen. Wie Sie in Kapitel 9 gesehen haben, muss jede Seite der Website:

- die PHP-Funktion `session_start()` aufrufen
- vor dem Zugriff prüfen, ob das superglobale Array `$_SESSION` die Daten enthält, auf die die Seite zuzugreifen versucht (sonst kann es zu einem `Undefined index`-Fehler kommen)

Anstatt den entsprechenden Code auf jeder Seite zu wiederholen, wird in der Datei `bootstrap.php` (die in jeder Seite enthalten ist) ein Session-Objekt mithilfe der neuen Session-Klasse (siehe unten) erstellt. Dieser Code ist in der `__construct()`-Methode enthalten, sodass er bei der Erstellung des Objekts ausgeführt wird. Wenn sich der Benutzer angemeldet hat, werden seine Daten aus dem superglobalen Array `$_SESSION` übernommen und in den Eigenschaften des Session-Objekts gespeichert.

13. Das Session-Objekt wird in `bootstrap.php` erstellt.
14. Seine Eigenschaften werden in einer globalen Twig-Variablen gespeichert, sodass sie von jedem Template abgerufen werden können.

```

①  [{% extends 'layout.html' %}]
②  [{% block title %}Log In{% endblock %}]
③  [{% block description %}Log in to your Creative Folk account{% endblock %}]
④  [{% block content %}]
    <main class="container" id="content">
⑤      <form method="post" action="login.php" class="form-membership">
          <section class="header"><h1>Log in:</h1></section>
⑥      {% if success %}<div class="alert alert-success">{{ success }}</div>{% endif %}
      {% if errors %}<div class="alert alert-danger">{{ errors.message }}</div>{% endif %}

          <label for="email">Email: </label>
          <input type="text" name="email" id="email" value="{{ email }}" class="form-control">
          <div class="errors">{{ errors.email }}</div>
          <label for="password">Password: </label>
          <input type="password" name="password" id="password" class="form-control">
          <div class="errors">{{ errors.password|raw }}</div>
          <input type="submit" class="btn btn-primary" value="Log in"><br>
          <p><a href="password-lost.php">Lost password?</a></p>
      </form>
    </main>
  {% endblock %}

```

```

⑦ public function login(string $email, string $password)
{
    $sql = "SELECT id, forename, surname, joined, email, password, picture, role
           FROM member
           WHERE email = :email;";           // SQL zum Abruf von Mitgliederdaten
⑨     $member = $this->db->runSQL($sql, [$email])->fetch(); // SQL ausführen
⑩     if (!$member) {                      // Wenn kein Mitglied gefunden wird
        return false;                     // false zurückgeben
    }                                     // Sonst
⑪     $authenticated = password_verify($password, $member['password']); // Übereinstimmung?
⑫     return ($authenticated ? $member : false); // Rückgabe member or false
}

```

```

$loader = new Twig\Loader\FilesystemLoader(APP_ROOT . '/templates'); // Twig-Loader
$twig   = new Twig\Environment($loader, $twig_options); // Twig-Umgebung
$twig->addGlobal('doc_root', DOC_ROOT); // Dokumentstamm
⑬ $session = $cms->getSession(); // Sitzung erstellen
⑭ $twig->addGlobal('session', $session); // Sitzung hinzufügen

```

SESSIONS ZUM SPEICHERN VON BENUTZERDATEN VERWENDEN

Die Kopfzeile, die auf jeder Seite der Website verwendet wird, muss wissen, ob ein Benutzer eingeloggt ist oder nicht:

- Ist dies der Fall, enthält die Kopfzeile einen Link zur Mitgliederseite mit dem Vornamen des Mitglieds als Linktext und seiner Kennung im Abfragestring.
- Wenn er nicht angemeldet ist, enthält die Kopfzeile Links zu den Seiten `login.php` und `register.php`.

Ist der eingeloggte Benutzer Administrator, zeigt die Kopfzeile auch einen Link zu den Administrationsseiten.

Um diese Links zu erstellen, muss jede Seite die ID, den Vornamen und die Rolle des Mitglieds kennen; daher werden diese Informationen in einer Session gespeichert.

Wie auf der vorherigen Seite gezeigt, erstellt `bootstrap.php` (die in jeder Seite enthalten ist) ein Objekt mit der rechts abgebildeten, benutzerdefinierten Session-Klasse. Die Verwendung einer solchen Session-Klasse fasst den Code für das Erstellen, den Zugriff, das Aktualisieren und Löschen von und auf Sessions an einer Stelle zusammen.

Diese Klasse hat drei Eigenschaften:

- `id` enthält die Kennung des Mitglieds.
- `forename` enthält den Vornamen des Mitglieds.
- `role` enthält die Rolle des Mitglieds.

Wenn das Objekt erstellt wird, bewirkt die Methode `__construct()` Folgendes:

- `session_start()` aufrufen
- prüfen, ob eine Session Details für dieses Mitglied enthält

Wenn ja, werden diese Werte in den Objekteigenschaften gespeichert. Ist dies nicht der Fall, werden die Eigenschaften mit Standardwerten belegt.

1. Der Namespace für die Klasse wird deklariert.
2. Die Klasse wird `Session` genannt.
3. Die drei Eigenschaften werden deklariert. Sie speichern die ID, den Vornamen und die Rolle eines Mitglieds.
4. Die Methode `__construct()` wird automatisch ausgeführt, wenn ein Session-Objekt mit dieser Klasse erstellt wird.
5. Die PHP-Funktion `session_start()` aktiviert Sessions und erneuert bestehende Sessions.
6. Wenn das superglobale Array `$_SESSION` Schlüssel mit den Namen `id`, `forename` und `role` enthält, werden deren Werte in den Eigenschaften des Session-Objekts gespeichert. Wenn nicht, werden in diesen Eigenschaften Standardwerte gespeichert.
7. Die Methode `create()` wird aufgerufen, wenn sich ein Benutzer anmeldet. Sie benötigt das Array, das die Daten des Mitglieds enthält.
8. Die PHP-Funktion `session_regenerate_id()` (S. 340) aktualisiert die Session-ID, die sowohl in der Session-Datei als auch im Cookie verwendet wird.
9. Die ID, der Vorname und die Rolle des Mitglieds werden dem superglobalen Array `$_SESSION` hinzugefügt.
10. Die `update()`-Methode ruft die `create()`-Methode auf. Da der Code zum Erstellen oder Aktualisieren der Session denselben Code erfordert, sollte er nicht wiederholt werden. Die `update()`-Methode wird als Alias der `create()`-Methode bezeichnet, da sie ein alternativer Name für den Aufruf der Anweisungen in der `create()`-Methode ist.
11. Die `delete()`-Methode wird aufgerufen, wenn der Benutzer auf den Logout-Link in der Navigation klickt. Ihre Aufgabe ist es, die Session zu beenden (zur Funktionsweise siehe S. 343).

```
<?php  
① namespace PhpBook\CMS;  
  
② class Session  
③ {  
    public $id; // Session-Klasse definieren  
    public $forename; // Mitglied-ID speichern  
    public $role; // Vornamen des Mitglieds speichern  
    // Rolle des Mitglieds speichern  
  
④     public function __construct()  
⑤     {  
        session_start(); // Start oder Neustart der Sitzung  
        $this->id = $_SESSION['id'] ?? 0; // id-Eigenschaft dieses Objekts festlegen  
        $this->forename = $_SESSION['forename'] ?? ''; // forename-Eigenschaft  
        $this->role = $_SESSION['role'] ?? 'public'; // role-Eigenschaft  
    }  
  
    // Neue Sitzung erstellen - auch Aktualisieren einer bestehenden  
⑦    public function create($member)  
    {  
        session_regenerate_id(true); // Sitzung-ID aktualisieren  
        $_SESSION['id'] = $member['id']; // Member-ID hinzufügen  
        $_SESSION['forename'] = $member['forename']; // Vorname hinzufügen  
        $_SESSION['role'] = $member['role']; // Rolle hinzufügen  
    }  
  
    // Vorhandene Sitzung aktualisieren - Alias für create()  
⑩    public function update($member)  
    {  
        $this->create($member);  
    }  
  
    // Vorhandene Sitzung löschen  
⑪    public function delete()  
    {  
        $_SESSION = []; // Leeres superglobales Array $_SESSION  
        $param = session_get_cookie_params(); // Session-Cookie-Parameter abrufen  
        setcookie(session_name(), '', time() - 2400, $param['path'], $param['domain'],  
                 $param['secure'], $param['httponly']); // Session-Cookie löschen  
        session_destroy(); // Session beenden  
    }  
}
```

DIE NAVIGATIONSLEISTE PERSONALISIEREN

Das in bootstrap.php erstellte Session-Objekt wird in einer globalen Twig-Variablen gespeichert, sodass jedes Template auf ihre Eigenschaften zugreifen kann.

1. In layout.html prüft eine if-Anweisung, ob die id-Eigenschaft des Session-Objekts den Wert 0 hat (was bedeutet, dass der Benutzer nicht eingeloggt ist).
2. Ist dies der Fall, zeigt das Template Links zum Anmelden und Registrieren an.
3. Andernfalls ist das Mitglied eingeloggt.

4. Ein Link zur Profilseite wird erstellt, wobei der Vorname des Mitglieds im Linktext angezeigt wird.
5. Eine if-Anweisung prüft, ob die Eigenschaft role des Session-Objekts den Wert admin hat. Wenn ja, wird ein Link zum Adminbereich angezeigt.
6. Ein Link zu logout.php ermöglicht es den Benutzern, sich abzumelden.
(Die Datei logout.php befindet sich im Code-Download; sie ruft lediglich die delete()-Methode des Session-Objekts auf und leitet den Benutzer dann zur Startseite weiter.)

c16/templates/layout.html

Twig

```
①  {% if session.id == 0 %}  
②  [   <a href="login.php" class="nav-item nav-link">Log in</a> /  
③  [   <a href="register.php" class="nav-item nav-link">Register</a>  
④  {  
⑤  [   <a href="member.php?id={{ session.id }}">{{ session.forename }}</a> /  
⑥  [   {  
⑦  [     <a href="admin/index.php">Admin</a> /  
⑧  [     {  
⑨  [       <a href="logout.php">Logout</a>  
⑩  {  
⑪  }
```

Ergebnis

Log in / Register

Print / Digital · Illustration / Photography 

Ivy / Admin / Logout

Print / Digital , Illustration / Photography 

OPTIONEN ZUR PROFILSEITE HINZUFÜGEN

member.php zeigt das Profil des Website-Mitglieds und Zusammenfassungen seiner Arbeiten an. Sieht sich ein eingeloggtes Mitglied sein Profil an, werden ihm Links zur Aktualisierung seines Profils und zum Hinzufügen neuer Arbeiten angezeigt.

1. In member.html prüft eine Twig-if-Anweisung, ob die (in der Session gespeicherte) ID des Mitglieds mit der ID des Mitglieds übereinstimmt, dessen Arbeit angezeigt wird. Wenn ja, werden neue Links unter dem Profil angezeigt.

2. In article-summaries.html prüft eine weitere Twig-if-Anweisung, ob die ID des Mitglieds, das sich die Seite ansieht, mit der ID des Mitglieds übereinstimmt, das den jeweiligen Artikel geschrieben hat. Wenn ja, wird ein Link zum Bearbeiten des Werks hinzugefügt. Die Datei work.php, die es Nutzern ermöglicht, Arbeiten hochzuladen, befindet sich im Code-Download. Sie ähnelt der Datei article.php im Adminbereich, allerdings ist ein Bild erforderlich, das Mitglied ist der Autor und es gibt keine Veröffentlichungsoption.

Twig

```
①  {% if session.id == member.id %}  
    <nav class="member-options">  
      <a href="work.php" class="btn btn-primary">Add work</a>  
      <a href="member-edit-profile.php" class="btn btn-primary">Edit profile</a>  
      <a href="member-edit-picture.php" class="btn btn-primary">Profile picture</a>  
    </nav>  
  {% endif %}
```

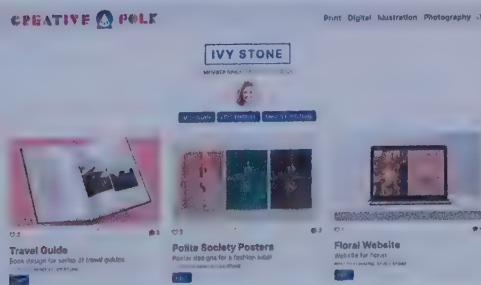
c16/templates/member.html

Twig

```
②  {% if session.id == article.member_id %}  
    <a href="work.php?id={{ article.id }}" class="btn btn-primary">Edit</a>  
  {% endif %}
```

c16/templates/article-summaries.html

Ergebnis



Quelle: CreativeFolk

Die Seiten zum Bearbeiten eines Profils und zum Hochladen oder Löschen von Profilbildern finden Sie im Code-Download. Sie können versuchen, diese Seiten zu schreiben, um Ihre Fähigkeiten zu testen:

Die Seite zum Bearbeiten von Profilen entspricht der zum Bearbeiten einer Kategorie. Der Code zum Hinzufügen/Löschen von Profilbildern entspricht dem zum Hinzufügen oder Löschen von Bildern in Artikeln.

ZUGRIFF AUF ADMIN-SEITEN EINSCHRÄNKEN

In den vorherigen Kapiteln konnte jeder auf die Admin-Seiten der Website zugreifen. In diesem Kapitel können nur Mitglieder mit der Rolle Admin auf diese Seiten zugreifen.

1. Jede Admin-Seite ruft die neue Funktion `is_admin()` auf, gleich nachdem die Datei `bootstrap.php` eingebunden wurde. Sie benötigt die Rolle des Mitglieds als Argument. (Wenn ein Mitglied nicht eingeloggt ist, setzt das Session-Objekt die Rolle auf `public`.)
2. Die Definition der Funktion `is_admin()` wird der Datei `functions.php` hinzugefügt.

3. Eine `if`-Anweisung prüft, ob die Rolle *nicht* admin ist.

4. Wenn nicht, wird der Benutzer auf die Startseite weitergeleitet. (Er wird zur Startseite und nicht zu `login.php` weitergeleitet, um zu verhindern, dass Personen, die keine Administratoren sind, die URLs der Admin-Seiten erraten können.)

5. Der `exit`-Befehl verhindert, dass weiterer Code auf der Seite, die die Funktion aufgerufen hat, ausgeführt wird. (Wenn der Benutzer Administrator ist, läuft der Rest der Seite weiter.)

c16/public/admin/article.php

PHP

```
<?php
// Part A: Setup
declare(strict_types = 1);
use PhpBook\Validate;

include '../../src/bootstrap.php';
① is_admin($session->role);

// Strikte Typisierung
// Import Validate-Namespace
// Setup-Datei einbinden
// Prüfen, ob Admin
```

c16/src/functions.php

PHP

```
② function is_admin($role)
{
    ③ if ($role != 'admin') {
        header('Location: ' . DOC_ROOT);
        ④ exit;
    }
}
```

Normalerweise sollte sich der Benutzer wie in diesem Kapitel anmelden, bevor er Aufgaben ausführen kann, die die Datenbank aktualisieren.

In seltenen Fällen sollen die Benutzer die Datenbank aktualisieren, ohne sich anzumelden, aber dies erfordert zusätzliche Sicherheitsmaßnahmen, wie Sie im Folgenden erfahren werden.

DATENBANK UND TOKEN DURCH E-MAIL-LINKS AKTUALISIEREN

Auf manchen Websites können Benutzer die Datenbank aktualisieren, ohne sich anzumelden, zum Beispiel, wenn sie eine E-Mail mit einem Link erhalten haben, etwa zum Zurücksetzen des Passworts. Die Links verwenden ein Token, um den Benutzer zu identifizieren.

Wenn ein Benutzer sein Kennwort vergisst, kann er sich nicht bei der Website anmelden, um es zurückzusetzen, also muss die Website ihm eine andere Möglichkeit bieten, es sicher zu aktualisieren.

Eine Lösung besteht darin, dem Benutzer per E-Mail einen Link zu einer Seite zu schicken, über die er das Passwort aktualisieren kann. Da der Link an die E-Mail-Adresse des Benutzers geschickt wird, sollte nur er in der Lage sein, den Link zu benutzen.

Der Link sollte zur Identifikation des Nutzers nicht die E-Mail-Adresse eines Mitglieds oder die ID aus der member-Tabelle verwenden, da ein Hacker den Link zu dieser Seite verwenden und die E-Mail-Adressen oder IDs anderer Mitglieder erraten könnte (und dann die Passwörter dieser Benutzer zurücksetzen und sich in deren Konten einloggen kann).

Stattdessen wird beim Zurücksetzen des Passworts ein Token zur Identifizierung des Benutzers erstellt. Dies ist ein zufälliger, eindeutiger Satz von Zeichen, der nicht leicht erraten werden kann, z. B:

0d9781153ed42ea7d72b4a4963dbd4f7f-bc1d09bca10 a8faae55d5dd66441521881a4e51eb17cd62596b156f 11218d31436e-5ae3381bcb50acbf31dd2c5cd197

Dieses Token wird

- in der Datenbank in der neuen Tabelle token gespeichert
- zur Identifizierung des Benutzers verwendet

Wenn der Benutzer auf den Link mit dem Token klickt, prüft die Website in der token-Tabelle der Datenbank (siehe S. 626), für welches Mitglied es erstellt wurde.

Auf den nächsten Seiten sehen Sie, wie Token verwendet werden, wenn ein Mitglied sein Passwort zurücksetzen möchte.

Zunächst geben die Nutzer ihre E-Mail in password-lost.php ein.

FORGOT PASSWORD?

Enter your email address:

SEND EMAIL TO RESET PASSWORD

Wird dieses Formular abgeschickt, prüft die Website, ob es einen Benutzer mit der angegebenen Mail-Adresse gibt. In diesem Fall wird ein neues Token in die token-Tabelle der Datenbank eingefügt und per E-Mail ein Link zur Seite password-reset.php gesendet, über die der Benutzer sein Passwort aktualisieren kann. Über das Token wird im Link das Mitglied identifiziert.

RESET PASSWORD

Enter Your New Password:

Confirm Your Password:

SUBMIT NEW PASSWORD

Wenn der Benutzer sein Passwort aktualisiert, wird mit der neuen Methode der member-Klasse passwordUpdate() nun das Passwort aktualisiert.

TOKEN IN DER DATENBANK SPEICHERN

Mit einer neuen Klasse Token wird ein Token-Objekt erstellt, das die Token erzeugt, sie in der neuen Token-Tabelle in der Datenbank speichert und die ID der Mitglieder zurückgibt, für die sie erstellt wurden.

Die neue Token-Tabelle muss mindestens den Token und die ID des Mitglieds speichern, für das er erstellt wurde. Für zusätzliche Sicherheit speichert jeder Token

- seine Verfallszeit, um zu verhindern, dass das Token lange nach der beabsichtigten Verwendung gültig bleibt. Für diese Website beträgt sie vier Stunden nach der Erstellung des Tokens.
- seinen Zweck: Eine Website kann Token für verschiedene Aufgaben verwenden; durch die Speicherung des Zwecks kann die Website überprüfen, ob ein Token wie gewünscht verwendet wird.

Ein weiterer beliebter Grund für die Verwendung von Token ist die Registrierung von Nutzern. Diese können per E-Mail einen Link erhalten, den sie anklicken müssen, bevor sie sich anmelden können. Damit wird bestätigt, dass die E-Mail-Adresse korrekt war.

Mit einer Klasse Token wird ein Token-Objekt mit zwei Methoden erstellt:

- `create()` erstellt ein neues Token und speichert es in der Datenbank.
- `getMemberId()` prüft, ob das Token noch nicht abgelaufen ist und ob es für den richtigen Zweck verwendet wird; wenn ja, gibt es die ID des Mitglieds zurück, für das es erstellt wurde.

Das Token-Objekt wird mit der neuen Methode `getToken()` des CMS-Objekts erstellt. Es wird in einer Eigenschaft des CMS-Objekts `token` gespeichert (dies spiegelt die Art und Weise wider, wie die Objekte Article, Category und Member erstellt werden).

token	member_id	expires	purpose
a730730065407fa0a0508cc7f06930ed962...	4	2021-03-08 14:04:01	password_reset
4fbba7d3ebd4c0f3269ef669e4123cc8a2d...	12	2021-03-08 14:05:09	password_reset
ba5fde0992dfc85b39397bf4df89ecaa25d...	9	2021-03-08 14:05:38	password_reset

Das Token besteht aus 64 zufälligen Zeichen.

Es wird mit zwei PHP-Methoden erzeugt:

- `random_bytes()` erzeugt eine Zeichenkette aus Zufallsbytes; ihr Parameter ist die Anzahl der zurückzugebenden Bytes
- `bin2hex()` wandelt binäre Daten in hexadezimale um

64 ZUFALLSBYTES ERZEUGEN

`bin2hex(random_bytes(64));`

BINÄR- IN HEXADEZIMALDATEN UMWANDELN

```

<?php
namespace PhpBook\CMS;
class Token
{
    protected $db; // Namespace deklarieren
    // Verweis auf Datenbankobjekt speichern

    public function __construct(Database $db)
    {
        $this->db = $db; // Datenbankobjekt in der Eigenschaft $db speichern
    }

    public function create(int $id, string $purpose): string
    {
        $arguments['token'] = bin2hex(random_bytes(64)); // Token
        $arguments['expires'] = date('Y-m-d H:i:s', strtotime('+4 hours')); // Ablauf
        $arguments['member_id'] = $id; // Member-ID
        $arguments['purpose'] = $purpose;
        $sql = "INSERT INTO token (token, member_id, expires, purpose)
                VALUES (:token, :member_id, :expires, :purpose);"; // SQL
        $this->db->runSQL($sql, $arguments); // SQL ausführen
        return $arguments['token']; // Token zurückgeben
    }

    public function getMemberId(string $token, string $purpose)
    {
        $arguments = ['token' => $token, 'purpose' => $purpose,]; // Token und Zweck
        $sql = "SELECT member_id FROM token WHERE token = :token
                AND purpose = :purpose AND expires > NOW();"; // Abruf der ID
        return $this->db->runSQL($sql, $arguments)->fetchColumn(); // Rückgabe ID / false
    }
}

```

1. Das Objekt muss mit der Datenbank arbeiten, daher speichert die Methode `__construct()` einen Verweis auf das Datenbankobjekt in der Eigenschaft `$db`.
2. Die `create()`-Methode erstellt ein neues Token und speichert es in der Token-Tabelle der Datenbank.
3. Ein Token wird erstellt und im Array `$arguments` gespeichert, bereit zum Hinzufügen zu einer SQL-Anweisung.
4. Das Datum und die Uhrzeit, zu der das Token abläuft (4 Stunden nach seiner Erstellung), wird dem Array `$arguments` hinzugefügt.
5. Die Mitgliedsnummer und der Zweck werden dem Array hinzugefügt.
6. `$sql` enthält den SQL-Code zum Hinzufügen eines Tokens zur Datenbank.

7. Der SQL-Code wird ausgeführt.
8. Das neue Token wird von der Methode zurückgegeben.
9. `getMemberId()` prüft, ob ein Token gültig ist. Ist dies der Fall, wird die Mitgliedsnummer zurückgegeben, andernfalls `false`.
10. Das Token und sein Zweck werden in einem Array gespeichert.
11. Die SQL-Abfrage versucht, eine Zeile in der Token-Tabelle zu finden, die das angegebene Token und den Zweck enthält und bei der die Ablaufzeit in der Zukunft liegt. Wenn sie eine Übereinstimmung findet, wird die Mitglieds-ID abgerufen.
12. Die SQL-Anweisung wird ausgeführt. Wenn eine Zeile übereinstimmt, gibt sie die ID des Mitglieds zurück. Wenn nicht, wird `false` zurückgegeben.

ANFRAGE, EIN PASSWORT ZURÜCKZUSETZEN

Die Seite password-lost.php (rechts) zeigt ein Formular an, in das die Benutzer eine E-Mail-Adresse eingeben und einen Link zur Aktualisierung des Passworts anfordern können. Nach dem Absenden

- überprüft die Seite, ob der Benutzer ein Mitglied ist, und ruft seine Kennung ab
 - erstellt die Seite ein Token, mit dem das Passwort zurückgesetzt werden kann, und speichert das Token in der Datenbank
 - erstellt und versendet die Seite eine E-Mail mit einem Link zur Seite, die das Passwort zurücksetzt
- 1.** Strikte Typisierung wird aktiviert, die Klasse Validate importiert, die Datei bootstrap.php eingebunden, und zwei Variablen, die die Twig-Seiten verwenden, werden initialisiert.
- 2.** Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
- 3.** Wenn ja, wird die E-Mail-Adresse erfasst und überprüft. Ist die E-Mail gültig, speichert die Variable \$error einen Leerstring, sonst eine Fehlermeldung.
- 4.** Eine `if`-Anweisung prüft, ob \$error eine leere Zeichenkette ist.
- 5.** Ist dies der Fall, wird der neuen Methode getIdByEmail() der Member-Klasse (siehe unten) die E-Mail-Adresse übergeben.
- Sie versucht, die E-Mail-Adresse in der Datenbank zu finden. Findet sie die E-Mail, gibt sie die ID des Mitglieds zurück, sonst false. Der Wert wird in \$id gespeichert.
- 6.** Eine weitere `if`-Anweisung prüft, ob eine ID gefunden wurde.

7. Ist dies der Fall, wird die Methode create() eines Token-Objekts aufgerufen, um ein neues Token für das Mitglied zu erstellen.

Der Zweck des Tokens wird auf password_reset gesetzt. Das neue Token, das zurückgegeben wird, wird in \$token gespeichert.

8. Es wird ein Link auf die Seite password-reset.php erstellt. Er enthält das neue Token in dem Abfragestring. Um diesen Link zu erstellen, muss die Website den Domänennamen der Website kennen. Dieser wird in der neuen Konstanten DOMAIN gespeichert, die in der Datei config.php angegeben wird.

Wenn noch nicht geschehen, öffnen Sie diese Datei und fügen Ihren Hostnamen (siehe S. 190) zu dieser Konstanten hinzu.

9. Der Betreff und der Text der E-Mail werden erstellt.

10. Ein neues Mail-Objekt wird mithilfe der Mail-Klasse erstellt (siehe S. 598–599); dann wird die E-Mail versendet. Wenn dies funktioniert, wird in der Variablen \$sent der Wert true gespeichert.

11. Die von Twig benötigten Informationen werden im Array \$data gespeichert, und die Methode render() von Twig wird aufgerufen.

12. Das Template password-lost.html (im zweiten Code-Kasten rechts) erstellt das Formular.

Eine Twig `if`-Anweisung prüft, ob die Variable sent den Wert false enthält. Ist dies der Fall, wird dem Benutzer ein Formular angezeigt, mit dem er einen Link zum Zurücksetzen des Passworts anfordern kann, wenn nicht, wird eine Meldung angezeigt, die besagt, dass der Benutzer Anweisungen zum Zurücksetzen seines Passworts per E-Mail erhalten hat.

c16/src/classes/CMS/Member.php

PHP

```
public function getIdByEmail(string $email)
{
    $sql = "SELECT id FROM member
           WHERE email = :email;";
    return $this->db->runSQL($sql, [$email])->fetchColumn(); // Abruf der Mitglieder-ID
} // Rückgabe der Mitglieder-ID
```

```

<?php
declare(strict_types = 1);
use PhpBook\Validate\Validate;
① include '../src/bootstrap.php';
$error = false;
$sent = false;

② if ($_SERVER['REQUEST_METHOD'] == 'POST') {           // Wenn Formular abgeschickt
    $email = $_POST['email'];                           // E-Mail
    ③ $error = Validate::isEmail($email) ? '' : 'Please enter your email'; // validieren
    ④ if ($error == '') {                                // Wenn valide
        ⑤ $id = $cms->getMember()->getIdByEmail($email); // Member-ID
        ⑥ if ($id) {                                     // ID gefunden
            ⑦ $token = $cms->getToken()->create($id, 'password_reset'); // Token
            ⑧ $link = DOMAIN . DOC_ROOT . 'password-reset.php?token=' . $token; // Link
            ⑨ $subject = 'Reset Password Link';           // Betreff + Body
            ⑩ $body = 'To reset password click: <a href="' . $link . '">' . $link . '</a>';
            $mail = new \PhpBook\Email\Email($email_config); // E-Mail-Objekt
            $sent = $mail->sendEmail($mail_config['admin_email'], $email,
                $subject, $body);                          // E-Mail senden
        }
    }
    ⑪ $data['navigation'] = $cms->getCategory()->getAll(); // Navigationskategorien
    $data['error']       = $error;                         // Validierungsfehler
    $data['sent']        = $sent;                          // Gesendet
}
echo $twig->render('password-lost.html', $data);          // Vorlage darstellen

```

```

⑫ {% extends 'layout.html' %}
{% block title %}Password Reset{% endblock %}
{% block content %}...
    {% if sent == false %}
        <form method="post" action="password-lost.php" class="form-membership"> ...
            <label for="email">Enter your email address: </label>
            <input type="text" name="email" id="email" class="form-control"><br>
            <input type="submit" name="submit" value="Send email to reset password" class="btn">
            <span class="errors">{{ error }}</span><br>
        </form>
    {% else %}
        <p>If your address is registered, we will email instructions to reset your password.</p>
    {% endif %}
    {% endblock %}

```

PASSWORT ZURÜCKSETZEN

Wenn ein Nutzer auf den Link in der E-Mail mit dem verlorenen Passwort klickt, wird er zur Datei `password-reset.php` (rechts) weitergeleitet. Findet sie ein gültiges Token im Abfragestring, zeigt sie ein Formular zur Passwortaktualisierung an.

1. Enthält der Abfragestring ein Token, wird es in `$token` gespeichert. Wenn nicht, wird der Benutzer zu `login.php` weitergeleitet.
2. Die Methode `getMemberId()` des Token-Objekts versucht, die ID des Mitglieds zu ermitteln. Im Erfolgsfall wird diese in `$id` gespeichert.
3. Wenn keine ID zurückgegeben wird, wird der Benutzer zu `login.php` geschickt. Im Erfolgsfall kann das Formular angezeigt oder bearbeitet werden.
4. Eine `if`-Anweisung prüft, ob das Formular abgeschickt wurde.
5. Wenn ja, werden das Passwort und die Bestätigung abgerufen.
6. Die Werte werden validiert, um sicherzustellen, dass sie den Anforderungen an Passwörter entsprechen und beide Felder dasselbe Passwort enthalten. Alle Fehler werden in `$errors` gespeichert.
7. Alle Fehler werden als eine einzige Zeichenkette in `$invalid` zusammengefasst.
8. Wenn ein Fehler gefunden wurde, wird eine Nachricht im Schlüssel `message` des Arrays `$errors` gespeichert.

c16/src/classes/CMS/Member.php

PHP

```
⑯ public function passwordUpdate(int $id, string $password): bool
{
    $hash = password_hash($password, PASSWORD_DEFAULT); // Passwort-Hash
    $sql = 'UPDATE member
            SET password = :password
            WHERE id = :id;'; // Passwort aktualisieren
    $this->db->runSQL($sql, ['id' => $id, 'password' => $hash,]); // SQL ausführen
    return true; // true zurückgeben
}
```

9. Andernfalls wird eine neue Methode der Klasse `Member` mit der Bezeichnung `passwordUpdate()` aufgerufen, um das Kennwort des betreffenden Mitglieds zu aktualisieren (siehe unten).

10. Die Daten des Mitglieds werden abgerufen.
11. Mit den Daten des Mitglieds wird eine E-Mail erstellt und gesendet, in der das Mitglied darüber informiert wird, dass sein Passwort aktualisiert wurde.
12. Das Mitglied wird dann auf die Anmeldeseite weitergeleitet und erhält eine Erfolgsmeldung, dass das Passwort aktualisiert wurde.
13. Die von Twig benötigten Informationen werden in `$data` gespeichert.
14. Das Template `password-reset.html` erstellt das Formular. Es ist im Code-Download zu finden.
15. Die neue Methode `passwordUpdate()` der Klasse `Member` (siehe unten) benötigt die ID des Mitglieds und sein neues Passwort.
16. Es wird ein Hash des neuen Passworts erstellt.
17. Eine SQL-Anweisung aktualisiert den für dieses Mitglied gespeicherten Passwort-Hash, und die Funktion gibt `true` zurück.

```
<?php
declare(strict_types = 1); // Strikte Typisierung
use PhpBook\Validate\Validate; // Import class

include '../src/bootstrap.php'; // Datei einrichten
$errors = [];; // Initialize array

① [ $token = $_GET['token'] ?? '';
if (!$token) { // Get token
    redirect('login.php'); // If id not returned
}
② $id = $cms->getToken()->getMemberId($token, 'password_reset'); // Get member id
③ [ if (!$id) { // If no id
    redirect('login.php', ['warning' => 'Link expired, try again.']); // Redirect
}

④ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Wenn Formular abgeschickt
⑤ [ $password = $_POST['password']; // Get new password
    $confirm = $_POST['confirm']; // Get password confirm
    // Validate passwords and check they match
    $errors['password'] = Validate::isPassword($password)
        ? '' : 'Passwords must be at least 8 characters and have:<br>
            A lowercase letter<br>An uppercase letter<br>A number
            <br>And a special character';
    $errors['confirm'] = ($password == $confirm) // Invalid password
        ? '' : 'Passwords do not match'; // Password does not match
    $invalid = implode($errors); // Join error messages
}

⑥ [ if ($invalid) { // If password not valid
    $errors['message'] = 'Please enter a valid password.'; // Store error message
} else { // Otherwise
    $cms->getMember()->passwordUpdate($id, $password); // Update password
    $member = $cms->getMember()->get($id); // Get member details
    $subject = 'Password Updated'; // Create subject and body
    $body = 'Your password was updated on ' . date('Y-m-d H:i:s') .
        ' - if you did not reset the password, email ' . $email_config['admin_email']; // Create email object
    $email = new \PhpBook\Email\Email($email_config); // Create email object
    $email->sendEmail($email_config['admin_email'], $member['email'], $subject, $body);
    redirect('login.php', ['success' => 'Password updated']); // Redirect to login
}

⑦ }

⑧ [ $data['navigation'] = $cms->getCategory()->getAll(); // All categories for nav
⑨ [ $data['errors'] = $errors; // Errors array
    $data['token'] = $token; // Token
⑩ $data['token'] = $token; // Token
⑪ echo $twig->render('password-reset.html', $data); // Vorlage darstellen
```

ZUSAMMENFASSUNG

MITGLIEDSCHAFT

- › Um sich auf einer Website zu registrieren, muss ein Mitglied eine eindeutige Kennung (z.B. eine E-Mail-Adresse) und ein Passwort angeben, um sich zu authentifizieren.
- › Informationen über Mitglieder können in der Datenbank zwischen Ihren Besuchen auf der Website gespeichert werden.
- › Wenn ein Mitglied zurückkehrt und sich anmeldet, kann sich eine Session daran erinnern, dass es sich angemeldet hat, und die entsprechenden Daten für die Dauer dieses Besuchs speichern.
- › Anstatt die Passwörter der Mitglieder in der Datenbank zu speichern, wird stattdessen ein Hash des Passworts gespeichert.
- › Rollen bestimmen, was die Mitglieder tun dürfen.
- › Token können verwendet werden, um Benutzer ohne persönliche Daten wie E-Mail oder ID zu identifizieren.
- › Token sollten verwendet werden, wenn Benutzer die Datenbank aktualisieren dürfen, ohne sich anzumelden.

17

WEITERE
FUNKTIONEN
HINZUFÜGEN

In diesem Kapitel lernen Sie, wie Sie einer Website neue Funktionen hinzufügen. Die URLs werden geändert, um sie SEO-freundlich zu machen, und die Nutzerinnen und Nutzer können Artikel liken und kommentieren.

SEO-freundliche URLs helfen bei der Suchmaschinenoptimierung (SEO) der Website, weil in den URLs Schlüsselwörter verwendet werden, z.B. in Artikeltiteln oder Kategorienamen.

Zum Beispiel wird die URL der Artikelseite <https://eg.link/article.php?id=24> in diese mit dem Titel der Seite geändert: <https://eg.link/article/24/travel-guide>.

Sobald Sie gelernt haben, wie Sie die URLs der Beispielsite in diese neue Struktur umwandeln können, fügen Sie im zweiten Teil zwei neue Funktionen hinzu, die es eingeloggten Mitgliedern ermöglichen,

- ein Bild zu **liken** (ähnlich wie bei Facebook, Instagram und Twitter, wo Mitglieder Beiträge anderer Mitglieder liken können),
- einen Beitrag zu **kommentieren**, um ihre Meinung und ihr Feedback hinzuzufügen.

Gehen Sie beim Hinzufügen neuer Funktionen zu einer Website so vor:

- notieren, was die Benutzer und Benutzerinnen tun können
- festlegen, welche Daten in der Datenbank gespeichert werden müssen
- die Funktionalität im PHP-Code und in den Templates implementieren

Die Kenntnisse, die Sie in diesem letzten Abschnitt erlangen, können Sie auch bei der Entwicklung neuer Websites einsetzen.

localhost

by Admin Logout

CREATIVE POLE

Print · Digital · Illustration / Photography

Quilt

▼ 2

Aug 03, 2021

This cover design is for a book about silent movies and is based on the dialogue boards they used. The card stock for the cover is designed to look slightly old and dusty and uses rough-textured recycled materials to give the ink a nice matte effect.

POSTED IN PRINT BY ELLI HOWARD

Comments

Samira Mirza
23 14 pm - August 03, 2021

This is beautiful work!

Grace Jackson
21 36 pm - August 09, 2021

What typeface did you use for the title?

SEO-FREUNDLICHE URLs

Wenn Sie in der URL Wörter verwenden, die den Inhalt einer Seite beschreiben, verbessern Sie die Suchmaschinenoptimierung (SEO), und die Seiten werden in den Suchmaschinen besser gefunden. Außerdem sind die URLs dadurch leichter zu lesen.

Bisher wurde in diesem Buch in der URL für jede Seite der Beispieleite der Pfad zu der PHP-Datei angegeben, die ausgeführt werden sollte. Musste die Seite Daten aus der Datenbank abrufen, wurde die ID der Daten im Abfragestring angegeben.

Viele Websites verwenden beschreibende, SEO-freundliche URLs statt Dateipfade. Wenn ein Besucher diese beschreibenden URLs anfordert, wandelt die Website die URL in einen Dateipfad um und teilt dieser Datei mit, welche Daten sie anzeigen soll. Diese Technik ist als **URL-Rewriting** bekannt.

Es gibt verschiedene Möglichkeiten, SEO-freundliche URLs zu schreiben. Unten sehen Sie neben den alten URLs, die in früheren Kapiteln verwendet wurden, das neue SEO-freundliche Format, das in diesem Kapitel genutzt wird.

ALTE URL

<https://localhost/register.php>

<https://localhost/login.php>

<https://localhost/category.php?id=2>

<https://localhost/category.php?id=4>

<https://localhost/article.php?id=19>

<https://localhost/article.php?id=24>

<https://localhost/member.php?id=2>

<https://localhost/admin/article.php?id=24> <https://localhost/admin/article/24>

Die neuen SEO-freundlichen URLs enthalten bis zu drei Teile, die jeweils durch einen Schrägstrich voneinander getrennt sind:

1. Sie beginnen ebenfalls mit einem Pfad zur Datei, aber die Dateierweiterung .php wird weggelassen. Diese Dateinamen eignen sich für SEO-freundliche URLs, weil sie den Zweck der Seite beschreiben.
2. Wenn die alte URL einen Abfragestring enthielt, der die Kennung der aus der Datenbank abzurufenden Daten enthielt, wird als Nächstes ein Schrägstrich gefolgt von der Kennung der aus der Datenbank abzurufenden Daten eingefügt.
3. Den Artikel- und Kategoriseiten werden SEO-freundliche Namen hinzugefügt, damit Suchmaschinen diese Seiten indizieren können:
 - Artikelseiten verwenden den Artikeltitel.
 - Kategoriseiten verwenden den Kategorienamen.

NEUE URL

<https://localhost/register>

<https://localhost/login>

<https://localhost/category/2/digital>

<https://localhost/category/4/photography>

<https://localhost/article/19/forecast>

<https://localhost/article/24/travel-guide>

<https://localhost/member/2>

Da die neuen URLs keine Dateipfade mehr enthalten, wird jede Anfrage für eine PHP-Seite an die Datei index.php gesendet. Diese nimmt die SEO-freundliche URL, teilt sie an jedem Schrägstrich und speichert dann jeden Teil als separates Element in einem Array. Für die angeforderten Seiten speichern die Teile des Array

- die Datei, die die Anfrage verarbeiten soll
- die ID der Daten in der Datenbank (falls verwendet)
- einen SEO-freundlichen Begriff (falls einer hinzugefügt wurde)

PFAD	TEILE	
register	\$parts[0] = 'register';	Dies erzeugt ein Array mit nur einem Element. Die Datei index.php soll die Seite register.php zur Registrierung eines neuen Nutzers einfügen.
category/2/	\$parts[0] = 'category';	Dadurch wird ein Array mit drei Elementen erstellt:
digital	\$parts[1] = '2'; \$parts[2] = 'digital';	<ul style="list-style-type: none"> ● Die Datei category.php wird eingefügt. ● Die ID der Kategorie ist 2. ● Der Name ist digital (dies hilft bei der Suchmaschinen-optimierung).
article/15/	\$parts[0] = 'article';	Dadurch wird ein Array mit drei Elementen erstellt:
seascape	\$parts[1] = '15'; \$parts[2] = 'seascape';	<ul style="list-style-type: none"> ● Die Datei article.php sollte eingefügt werden. ● Die ID des Artikels ist 15. ● Der Titel ist seascape (dies hilft bei der SEO)

Suchmaschinen sollten die Admin-Seiten nicht indizieren, daher enden ihre URLs nicht mit SEO-freundlichen Begriffen. Wird die URL für eine Admin-Seite an jedem Schrägstrich geteilt und in ein Array verwandelt, wird signalisiert,

- dass es sich um eine Admin-Seite handelt,
- welche Datei die Anfrage verarbeiten soll,
- wie die ID der Daten lautet, die die Seite nutzt (falls erforderlich)

Als Nächstes fügt index.php dem Wert im ersten Element des Arrays die Dateierweiterung .php hinzu. Dadurch werden die in den alten URLs verwendeten Dateinamen repliziert. Die Seite index.php fügt dann diese Datei ein, um die Anfrage zu bearbeiten. Die folgende Tabelle zeigt:

- den Pfad (das ist der Teil der URL nach dem Host)
- das Array, das bei der Aufteilung des Pfads entsteht
- die Beschreibung, wofür jeder Teil verwendet werden soll

PFAD	TEILE	
admin/article/15	\$parts[0] = 'admin'; \$parts[1] = 'article'; \$parts[2] = '15';	Dieses Array gibt an: <ul style="list-style-type: none"> ● Die einzubindende PHP-Datei ist admin/article.php. ● Die ID des zu verwendenden Artikels ist 15.
admin/category/1	\$parts[0] = 'admin'; \$parts[1] = 'category'; \$parts[2] = '1';	Dieses Array gibt an: <ul style="list-style-type: none"> ● Die einzuschließende PHP-Datei ist admin/category.php. ● Die ID der zu verwendenden Kategorie ist 1.

HINWEIS: URLs dürfen keine Leerzeichen enthalten, und einige Zeichen haben auch eine besondere Bedeutung, wie z. B. / ? : ; @= & " < > # % { } | \ ^ ~ [] ` .

Sobald index.php das Array erstellt hat, wird geprüft, ob der Wert des ersten Elements admin ist. Wenn ja, handelt es sich um eine Anfrage für eine Admin-Seite, und der Pfad zu der einzubindenden Datei wird erstellt, indem die folgenden Elemente zusammengefügt werden:

- der Wert im ersten Element
- ein Schrägstrich
- der Wert im zweiten Element
- die Dateierweiterung .php

Diese Zeichen müssen aus dem Artikeltitel oder Kategorienamen entfernt werden, bevor sie in einer URL verwendet werden, und dieser neue Wert wird in der Datenbank gespeichert.

AKTUALISIERTE DATEISTRUKTUR

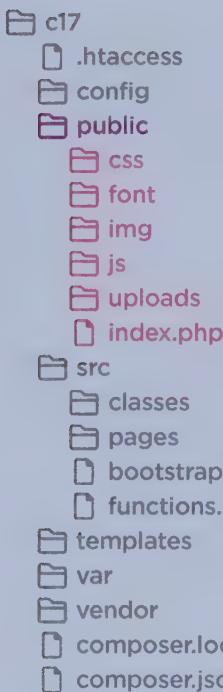
Die Datei `index.php` ist die einzige PHP-Datei, die sich noch im Stammverzeichnis des Dokuments befindet. Alle PHP-Seiten, die sie bei der Verarbeitung der URL einfügt, wurden über das Stammverzeichnis des Dokuments in das Verzeichnis `src/pages` verschoben.

Unten sehen Sie die neue Dateistruktur, die in diesem Kapitel verwendet wird.

Alle PHP-Seiten wurden aus dem Verzeichnis `public` (dem Stammverzeichnis des Dokuments) in das Verzeichnis `src/pages` verschoben.

Es gibt zwei neue Dateien im Ordner `public`:

- `.htaccess` enthält Regeln, die alle Anfragen an `index.php` leiten (es ist eine Konfigurationsdatei für Apache).
- `index.php` verarbeitet die URL und fügt die entsprechende PHP-Seite aus dem Ordner `src/pages` ein.



← ANWENDUNGSTMAMMVERZEICHNIS

← Regeln für URL

← DOKUMENTSTMAMMVERZEICHNIS

← URL verarbeiten und entsprechende Seiten einfügen

← PHP-Seiten

LEGENDE:

- Dokument-Root (dark purple circle)
- Im Dokument-Root (pink circle)
- Über Dokument-Root (grey circle)

SEO-FREUNDLICHE URLs IMPLEMENTIEREN

Wenn Sie neue Funktionen hinzufügen, müssen Sie überlegen, welche Daten Sie speichern, wie Sie die Funktionen im Code implementieren und wie Sie die Schnittstelle aktualisieren. Im Folgenden werden diese Fragen für das Hinzufügen von SEO-freundlichen URLs behandelt.

WELCHE DATEN SOLLEN WIE GESPEICHER WERDEN?

Wenn ein Artikel oder eine Kategorie erstellt oder aktualisiert wird, wird dafür ein SEO-freundlicher Name in der Datenbank gespeichert.

Die Namen werden in der Spalte `seo_title` der Tabelle `article` und in der Spalte `seo_name` der Tabelle `category` gespeichert. Unten sehen Sie die aktualisierte Tabelle `category`.

Kategorie				
	<code>id</code>	<code>Name</code>	<code>description</code>	<code>navigation</code>
1	Print	Inspiring graphic design		1
2	Digital	Powerful pixels		1
3	Illustration	Hand-drawn visual storytelling		1

NEUE FUNKTIONEN IM CODE IMPLEMENTIEREN

Da SEO-freundliche URLs keinen Pfad zu einer PHP-Datei verwenden, weisen Regeln in einer neuen `.htaccess`-Datei im Stammverzeichnis des Dokuments den Webserver an, alle Anfragen für PHP-Seiten an `index.php` zu senden. `index.php` verarbeitet dann die angeforderte URL und

- ruft die Kennung der benötigten Daten ab und speichert sie in einer Variablen,
- fügt die korrekte PHP-Datei für die Bearbeitung der Anfrage ein.

Wenn Artikel oder Kategorien hinzugefügt oder bearbeitet werden,

- erzeugt die neue Funktion `create_seo_name()` die SEO-freundlichen Namen,
- speichern vorhandene Methoden der Klassen `Article` und `Category` diese neuen Namen in der Datenbank.

Wenn Artikel- und Kategoriedaten aus der Datenbank abgerufen werden, wird der SEO-freundliche Name zurückgegeben und an die Twig-Templalte übergeben, um die neuen Links zu erstellen.

DIE SCHNITTSTELLE AKTUALISIEREN

In den Twig-Templates müssen alle Links aktualisiert werden. Wenn sie einen SEO-freundlichen Namen haben, wird dieser hinzugefügt.

```
<a href="article/{{ article.id }}/{{ article.seo_title }}">
```

Unten sehen Sie einen Link zu einer Artikelseite. Er verwendet die Artikel-ID und dann den SEO-freundlichen Titel.

URLS UMSCHREIBEN LASSEN

Der Apache-Webserver verfügt über eine eingebaute URL-Rewriting-Engine. Regeln bestimmen, wann eine Anfrage für eine URL in eine Anfrage für eine andere URL umgewandelt werden soll.

SEO-freundliche URLs werden nur für die Seiten verwendet, die von den Besuchern angefordert werden, etwa Artikel-, Kategorie- und Mitgliederseiten – und nicht für die übrigen von diesen Seiten verwendeten Dateien wie Bild-, CSS-, JavaScript- oder Schriftartdateien; ihre URLs bleiben gleich. Daher geht die URL-Rewriting-Engine des Apache-Webservers folgendermaßen vor:

- Bild-, CSS-, JavaScript- und Schriftartdateien wie bisher behandeln, da sich ihre URLs nicht geändert haben
- alle anderen Anfragen an die Datei `index.php` senden

Die Datei `index.php` verarbeitet die URL und fügt die entsprechende Datei ein.

Wie Sie auf S. 196–199 gesehen haben, steuern `.htaccess`-Dateien die Einstellungen für den Apache-Webserver (auch die der URL-Rewriting-Engine). Im Ordner `c17` befindet sich eine `.htaccess`-Datei mit Einstellungen für die Zeichenkodierung und Datei-Upload-Größen. Die Regeln zur Steuerung der URL-Rewriting-Engine werden in diese Datei eingefügt.

`c17/public/.htaccess`

PHP

```
① RewriteEngine On  
② RewriteCond %{REQUEST_FILENAME} !-f  
③ RewriteRule . public/index.php
```

Eine `.htaccess`-Datei kann mehrere Bedingungen enthalten, jeweils mit Regeln, die ausgeführt werden, wenn die Bedingung erfüllt ist..

1. Zunächst wird die URL-Rewriting-Engine eingeschaltet.

Die Anweisungen für die Rewriting-Engine bestehen aus zwei Teilen in separaten Zeilen:

- **Bedingung**, die angibt, wann eine Regel ausgeführt werden soll
- **Regel**, die angibt, was passiert, wenn die Bedingung erfüllt ist

2. Die Bedingung lautet: Wenn die Anfrage eine Datei betrifft, die auf dem Server *nicht* existiert, wird die nächste Regel ausgeführt:

Die URLs für Bilder, CSS-, JavaScript- und Schriftdateien verweisen auf den Speicherort dieser Dateien auf dem Server (sodass die nachfolgende Regel *nicht* ausgeführt wird).

SEO-freundliche URLs verweisen nicht auf eine Datei auf dem Server (daher *wird* die nachfolgende Regel für sie ausgeführt).

3. Die Regel gibt an, dass die Anfrage von der Datei `index.php` bearbeitet werden soll, die sich im Stammordner des Dokuments befindet.

Es gibt noch weitere leistungsstarke Tools zum Umschreiben von URLs, die in diesem Buch jedoch nicht alle behandelt werden können.

URLS AKTUALISIEREN

Bei der Verwendung von SEO-freundlichen URLs sollten Links zu anderen Seiten sowie Bild-, CSS-, JavaScript- und Schriftartdateien relativ zum Dokumentstamm sein. Normalerweise wird hier ein Schrägstrich verwendet, aber für die Beispielseite ist eine Konstante erforderlich.

In früheren Kapiteln wurden für Links zu anderen Seiten der Website und für Links zu Bild-, CSS-, JavaScript- und Schriftartdateien URLs verwendet, die relativ zur aktuellen PHP-Seite waren.

Durch die neuen SEO-freundlichen URLs scheinen sich die Seiten in verschiedenen Ordnern zu befinden. Diese beiden URLs scheinen zum Beispiel auf Ordner zu verweisen, die es jedoch gar nicht gibt:

```
/category/1/print  
/article/22/polite-society-posters
```

Daher müssen alle relativen Links aktualisiert werden, sodass die Pfade relativ zum Stammordner der Webseite und nicht zur aktuellen Seite sind.

Normalerweise wird ein Schrägstrich als Pfad zum Stammordner des Dokuments verwendet. Da der Code-Download jedoch mehrere Versionen der Beispielseite enthält, wird der Ordner `public` im Code der einzelnen Kapitel seit Kapitel 14 so behandelt, als wäre er der Stammordner des Dokuments. Der Pfad zum Ordner `public` wird **allen** relativen Links in der Site vorangestellt.

Dieser Pfad wird in `config.php` in einer Konstanten gespeichert. `bootstrap.php` fügt ihn zu den globalen Twig-Variablen `doc_root` hinzu, sodass er in allen Templates verwendet werden kann. Unten sehen Sie ihn am Anfang von Links zu Seiten (die jetzt SEO-freundliche Titel haben) und Bilddateien.

TWIG

```
{% for article in articles %}  
<article class="summary">  
  <a href="{{ doc_root }}article/{{ article.id }}/{{ article.seo_title }}">  
    {% if article.image_file %}  
        
    {% else %}  
        
    {% endif %}  
  
    ...  
  
    {% if session.id == article.member_id %}  
      <a href="{{ doc_root }}work/{{ article.id }}" class="btn btn-primary">Edit</a>  
    {% endif %}</article>  
  {% endfor %}
```

ANFRAGEN BEARBEITEN

Jede Anfrage, die nicht für eine Bild-, CSS-, JavaScript- oder Schriftartdatei bestimmt ist, wird an die Datei `index.php` gesendet. Diese verarbeitet die URL, wandelt sie in ein Array um und fügt dann die richtige PHP-Datei ein.

1. In die Datei `index.php` wird `bootstrap.php` eingefügt, damit diese Anweisung nicht auf allen PHP-Seiten wiederholt werden muss. Anschließend wird die angeforderte URL in ein Array umgewandelt.
2. Der angeforderte Pfad (der Teil nach dem Hostnamen) wird aus PHPs superglobaler Variablen `$_SERVER` entnommen. Er wird in Kleinbuchstaben umgewandelt und in `$path` gespeichert.
3. Der Pfadteil bis zum Ordner `public` für dieses Kapitel (der das Stammverzeichnis des Dokuments wäre) wird aus der Variablen entfernt. Für dieses Kapitel wäre das `phpbook/section_d/c17/public/` (dieser Schritt ist nur notwendig, weil der Download-Code mehrere Siteversionen enthält).
4. Mit der PHP-`explode()`-Funktion wird der Pfad an jedem Schrägstrich geteilt und jeder Teil als separates Element im Array `$parts` gespeichert.

Wie Sie auf S. 637 gesehen haben, gibt das erste Element die zu verwendende Datei an, wenn eine öffentliche Seite angefordert wird. Wenn es sich um eine Admin-Seite handelt, hat das erste Element den Wert `admin`, und das zweite Element gibt die zu verwendende Datei an. Zum Beispiel:

PFAD	TEILE
article/15/seascape	<code>\$parts[0] = 'article';</code> <code>\$parts[1] = '15';</code> <code>\$parts[2] = 'seascape';</code>
admin/article/15	<code>\$parts[0] = 'admin';</code> <code>\$parts[1] = 'article';</code> <code>\$parts[2] = '15';</code>

5. Die Datei prüft, ob es sich bei der Anfrage um eine öffentliche Seite oder eine Admin-Seite handelt, indem der Wert des ersten Elements im Array `$parts` geprüft wird. Ist das erste Element des Arrays nicht `admin`, bezieht sich die Anfrage auf eine öffentlich zugängliche Seite.
6. Das erste Element des `$parts`-Arrays bestimmt den Namen der Datei, die für die Bearbeitung der Anfrage eingebunden werden soll. Wenn der Benutzer zum Beispiel eine Artikelseite anfordert, wäre der Wert `article` (wie in der Tabelle unten links gezeigt).

Würde der Benutzer die Startseite anfordern, gäbe es keinen Wert für dieses Element des Arrays, sodass `$page` stattdessen `index` speichern soll.

Um diesen Wert zu speichern, wird eine neue Kurzform des ternären Operators, der sogenannte Elvis-Operator, verwendet.

Anstatt das Folgende zu schreiben:

`$page = $parts[0] ? $parts[0] : 'index';`
wird stattdessen diese Kurzform verwendet:

`$page = $parts[0] ?: 'index';`

Wenn `$parts[0]` einen Wert enthält, wird er in `$page` gespeichert, wenn nicht, speichert `$page` den Wert `index`.

7. Wenn das zweite Element des Arrays einen Wert enthält, ist das die ID der Daten, mit denen die Seite arbeitet. Befindet sich im zweiten Teil des Arrays eine ID, wird sie in `$id` gespeichert; andernfalls speichert `$id null`.

```

<?php
① include '../src/bootstrap.php';                                // Datei einrichten

② $path  = mb_strtolower($_SERVER['REQUEST_URI']);           // Pfad in Kleinbuchstaben
③ $path  = substr($path, strlen(DOC_ROOT));                  // bis DOC_ROOT entfernen
④ $parts = explode('/', $path);                            // bei / in Array teilen

⑤ if ($parts[0] != 'admin') {                                // Wenn Admin-Seite
⑥     $page = $parts[0] ?: 'index';                         // Seitenname (oder index)
⑦     $id   = $parts[1] ?? null;                           // ID (oder null)
⑧ } else {                                                 // Wenn keine Admin-Seite
⑨     $page_ = 'admin/' . ($parts[1] ?? '');             // Seitenname
⑩     $id   = $parts[2] ?? null;                           // ID ermitteln
}
⑪ $id = filter_var($id, FILTER_VALIDATE_INT);                // ID validieren

⑫ $php_page = APP_ROOT . '/src/pages/' . $page . '.php';    // Pfad zur PHP-Seite
⑬ if (!file_exists($php_page)) {                            // Wenn Seite nicht in Array
⑭     $php_page = APP_ROOT . '/src/pages/page-not-found.php'; // page not found einfügen
}
⑮ include $php_page;                                       // PHP-Datei einfügen

```

8. Wird eine Admin-Seite angefordert, enthält das erste Element im Array \$parts das Wort admin; das zweite den Namen der Seite.
 9. Die Variable \$page beginnt mit dem Aufbau des Pfads zu der Datei, die die Anfrage bearbeiten soll. Sie verwendet die Zeichenkette admin/, gefolgt von dem Namen der Seite. (Würde die URL mit admin/ enden (also keine Seite angeben), gäbe es parts[1] nicht, sodass der Null-Koaleszenz-Operator den Wert durch eine leere Zeichenkette ersetzt.)
 10. Enthält die URL eine ID, wird sie in \$id gespeichert; andernfalls wird \$id als Null gespeichert.
 11. Mit der PHP-Funktion filter_var() wird geprüft, ob der in \$id gespeicherte Wert eine ganze Zahl ist. Dadurch muss eine PHP-Seite, die eine id verwendet, diese Anweisung nicht wiederholen.
- Zu diesem Zeitpunkt enthält die Seite drei Variablen:
- \$parts: das Array der Teile der URL
 - \$page: der Name der Seite (wenn es sich um eine Admin-Seite handelt, steht vor dem Namen admin/)
 - \$id: die ID, falls in der URL eine angegeben wurde

12. Der Pfad zu der Seite, die die Anfrage bearbeiten soll, wird in \$php_page gespeichert. Dazu werden
 - der Wert in APP_ROOT (erstellt in bootstrap.php),
 - der Pfad /src/pages/ zu den PHP-Seiten,
 - der Wert in \$page und
 - die .php-Dateierweiterung
 zusammengefügt.
13. Die PHP-Funktion file_exists() prüft, ob der Pfad zu der in Schritt 12 erstellten PHP-Datei *nicht* mit einer echten Datei auf dem Server übereinstimmt.
14. Wenn er nicht übereinstimmt, wird der in \$php_page gespeicherte Wert mit dem Pfad zur Datei page-not-found.php aktualisiert.
Dies wird durch einen exit-Befehl beendet, der die Ausführung von weiterem Code stoppt.
15. Wenn die PHP-Seite noch läuft, wird die in \$php_page gespeicherte PHP-Datei in die Seite eingebunden. Die eingefügten PHP-Seiten werden genau so aufführt, wie sie in der URL angefordert wurden (als wäre der Code kopiert und an der Stelle eingefügt worden, an der sich die PHP-include-Anweisung befindet).

SEO-NAMEN ERSTELLEN

Wenn Artikel und Kategorien erstellt oder aktualisiert werden, wird mit der Funktion `create_seo_name()` ein SEO-freundlicher Name für den Artikel oder die Kategorie erstellt.

Da URLs keine Leerzeichen oder bestimmte Zeichen mit besonderer Bedeutung (wie `/` `=` `&`) enthalten dürfen, wurde der Datei `functions.php` die Funktion `create_seo_name()` hinzugefügt, die aus Artikeltiteln und Kategorienamen einen SEO-freundlichen Namen erstellt, der nur die Buchstaben A-z, die Zahlen 0-9 und Bindestriche enthält.

Die PHP-Funktion `transliterator_transliterate()` versucht zudem, Nicht-ASCII-Zeichen durch das nächstliegende ASCII-Äquivalent zu ersetzen (auf Basis phonetischer Ähnlichkeit). Zum Beispiel würde Über zu Über und École zu Ecole. Damit die Transliteration funktioniert, muss eine Apache-Erweiterung installiert sein. Daher verwendet der Code die PHP-Methode `function_exists()`, um zu prüfen, ob die Funktion verfügbar ist, bevor er sie aufruft. Nur dann führt er die Aufgabe aus. Mehr dazu unter: <http://notes.re/php/transliteration>.

1. `create_seo_name()` nimmt einen String als Parameter und gibt eine SEO-freundliche Version dieses Texts zurück.
2. Der Text wird in Kleinbuchstaben umgewandelt.
3. Alle Leerzeichen werden am Anfang und Ende entfernt.
4. Die in PHP eingebaute Funktion `function_exists()` prüft, ob die Funktion `transliterator_transliterate()` verfügbar ist. Wenn ja, wird sie aufgerufen, um Nicht-ASCII-Zeichen durch ASCII-Äquivalente zu ersetzen.
5. `preg_replace()` ersetzt Leerzeichen durch Bindestriche.
6. Anschließend wird alles außer -, A-z oder 0-9 entfernt.
7. Der aktualisierte Artikel- oder Kategorienname wird zurückgegeben.

c17/src/functions.php

```
PHP  
① function create_seo_name(string $text): string  
{  
    ②     $text = strtolower($text);                                // Text in Kleinbuchstaben  
    ③     $text = trim($text);                                     // Leerzeichen entfernen  
    ④     if (function_exists('transliterator_transliterate')) { // Wenn Transliterator installiert  
        $text = transliterator_transliterate('Latin-ASCII', $text); // Transliteration  
    }  
    ⑤     $text = preg_replace('/ /', '-', $text);                // Leerzeichen durch - ersetzen  
    ⑥     $text = preg_replace('/[^A-z0-9]+/', '', $text);         // Entfernen, wenn kein -, A-z oder 0-9  
    ⑦     return $text;                                         // SEO-Name zurückgeben  
}
```

SEO-NAMEN SPEICHERN

Die Funktion `create_seo_name()` wird aufgerufen, wenn ein Artikel oder eine Kategorie entweder erstellt oder aktualisiert wird. Dieser Name wird dann an die Methoden zur Aktualisierung der Datenbank weitergegeben.

1. Wenn eine Kategorie erstellt oder aktualisiert wird, ruft `category.php` (jetzt in `src/pages/admin`) `create_seo_name()` auf und über gibt ihr den Kategorienamen als Argument.

Der zurückgegebene SEO-freundliche Name wird im Kategoriedaten-Array gespeichert. Dieses Array wird dann an die Methoden `create()` oder `update()` des Category-Objekts übergeben.

PHP

```
$category['name']      = $_POST['name'];           // Name abrufen
$category['description'] = $_POST['description'];    // Beschreibung
$category['navigation'] = (isset($_POST['navigation'])) ? 1 : 0; // Navigation
① $category['seo_name'] = create_seo_name($category['name']); // SEO-freundlicher Name
```

`c17/src/pages/admin/category.php`

2. Wenn die Methoden `create()` oder `update()` des Category-Objekts aufgerufen werden, enthält das Array `$category` ein neues Element mit dem SEO-freundlichen Namen.

Der Prozess für Artikel ist derselbe:

Der Schlüssel `seo_title` wird dem Array `$article` in `admin/article.php` und `work.php` hinzugefügt. Der SEO-freundliche Titel wird in der Datenbank gespeichert, wenn die Methoden `create()` oder `update()` der Klasse `Article` aufgerufen werden.

PHP

```
② public function create(array $category): bool
{
    try {
        $sql = "INSERT INTO category (name, description, navigation, seo_name)
                VALUES (:name, :description, :navigation, :seo_name);"; // SQL
    }
    ...
}
```

`c17/src/classes/CMS/Category.php`

HINWEIS: Die Spalten `seo_name` und `seo_title` in der Datenbank haben Eindeutigkeitsbedingungen (wie die Spalten `name` und `title`), um sicherzustellen, dass die SEO-Namen eindeutig sind.

Den Code zum Speichern von SEO-freundlichen Namen finden Sie auch in Kapitel 16, um sicherzustellen, dass die Datenbank die Namen speichert, wenn Änderungen dieses Kapitelcodes vorgenommen werden.

SEITEN MIT SEO-FREUNDLICHEN NAMEN ANZEIGEN

Die Artikel- und Kategorienseiten verwenden SEO-freundliche Namen in den URLs. Diese Seiten prüfen, ob der SEO-freundliche Name korrekt ist, bevor die Seite angezeigt wird.

Zunächst werden die `get()`- und `getAll()`-Methoden der Klassen `Article` und `Category` aktualisiert, um ihre SEO-freundlichen Namen aus der Datenbank abzurufen.

1. Der SQL-Code in der `get()`-Methode der Kategorieklasse fordert Daten aus der Spalte `seo_name` an.

Als Nächstes werden zwei Aufgaben aus **allen** PHP-Dateien entfernt, die in den Ordner `src/pages` verschoben wurden, da in `index.php` nun die folgenden Aufgaben ausgeführt werden:

- Datei `bootstrap.php` einbinden
- eine ID aus dem Abfragestring abrufen und diese validieren

Dann wird eine neue Aufgabe zu den Dateien `article.php` und `category.php` hinzugefügt. Sie prüft, ob der SEO-freundliche Name in der URL korrekt ist, da mehrere Links auf denselben Artikel mit unterschiedlichen Titeln verweisen könnten, z. B.:

- ⊕ <http://eg.link/article/24/travel-guide>
- ⊕ <http://eg.link/article/24/japan-guide>
- ⊕ <http://eg.link/article/24/guide-book>

Jede dieser URLs enthält die Informationen, die die Website benötigt, um die Daten für die Seite abzurufen (den Typ der einzuschließenden Seite und eine ID). Suchmaschinen könnten sie jedoch für verschiedene Seiten mit doppeltem Inhalt halten, und dafür könnte die Website abgestraft werden. Dies könnte der Fall sein, wenn eine andere Website einen Link zu der Seite falsch geschrieben hat oder wenn sich der Artikeltitel geändert hat, nachdem der Link erstellt wurde.

Daher überprüfen die Seiten `article.php` und `category.php`, ob der SEO-freundliche Teil der URL (im Array `$parts` der Datei `index.php` gespeichert) mit dem SEO-freundlichen Namen in der Datenbank übereinstimmt. Ist dies nicht der Fall, wird der Benutzer auf die korrekte URL umgeleitet.

2. Eine `if`-Anweisung prüft, ob der SEO-freundliche Name aus der URL (der Wert im dritten Element des `$parts`-Arrays, das in `index.php` erstellt wurde) mit dem SEO-freundlichen Namen in der Datenbank übereinstimmt. (Beide Werte werden in Kleinbuchstaben umgewandelt, bevor sie verglichen werden.)

3. Wenn sie nicht übereinstimmen, schickt die `redirect()`-Funktion den Besucher auf dieselbe Seite mit dem korrekten SEO-freundlichen Namen in der URL.

Schließlich muss jeder Link in allen Templates so umgestellt werden, dass die SEO-freundlichen URLs verwendet werden.

4. Der Pfad setzt sich zusammen aus:

- dem Pfad zum Stammordner des Dokuments der Website (normalerweise ist dies ein `/`, aber da der Code-Download mehrere Versionen der Website enthält, ist es der Pfad zum Ordner `public` für dieses Kapitel)
- dem PHP-Seitennamen ohne die Erweiterung `.php`
- der Kennung für den Artikel oder die Kategorie
- einem SEO-Namen, wenn der Link zu einem Artikel oder einer Kategorie gehört

5. Der Pfad zu den Bilddateien muss auch den Pfad zum Stammverzeichnis des Dokuments enthalten (siehe S. 641).

PHP

c17/src/classes/CMS/Category.php

```

public function get(int $id)
{
    ①   $sql = "SELECT id, name, description, navigation, seo_name
              FROM category
              WHERE id = :id;";           // SQL für eine Kategorie
    return $this->db->runSQL($sql, [$id])->fetch(); // Kategoriedaten zurückg.
}

```

PHP

c17/src/pages/category.php

```

<?php
declare(strict_types = 1);                                // Strikte Typisierung
if (!$id) {                                               // Wenn keine valide ID
    include APP_ROOT . '/src/pages/page-not-found.php'; // Page not found
}

$category = $cms->getCategory()->get($id);             // Kategoriedaten
if (!$category) {                                         // Wenn Kategorie leer
    include APP_ROOT . '/src/pages/page-not-found.php'; // Page not found
}

② if (mb_strtolower($parts[2]) != mb_strtolower($category['seo_name'])) { // Falscher SEO-Name
③     redirect('category/' . $id . '/' . $category['seo_name'], [], 301); // Redirect
}

$data['navigation'] = $cms->getCategory()->getAll();      // Navigationskategorien
$data['category'] = $category;                            // Aktuelle Kategorie
$data['articles'] = $cms->getArticle()->getAll(true, $id); // Artikel
$data['section'] = $category['id'];                      // Kategorie-ID für Nav.

```

TWIG

c17/templates/article-summaries.html

```

④ <a href="{{ doc_root }}article/{{ article.id }}/{{ article.seo_title }}">
    {% if article.image_file %}
        
    {% else %}
        
    {% endif %}
    ...

```

NEUE FUNKTIONEN PLANEN

Wenn Sie eine neue Funktion hinzufügen, sollten Sie sich zunächst genau überlegen, was Ihre Benutzer damit tun können. Das vereinfacht die Entwicklung des Codes zur Implementierung dieser Funktion.

Bevor Sie mit der Programmierung neuer Funktionen für die Website beginnen, sollten Sie klar definieren, was die Benutzerinnen und Benutzer dort tun können. Dadurch wird klar, wie die Aufgabe aufgeteilt werden kann.

In diesem Kapitel ist zum Beispiel auf allen Seiten mit Artikelzusammenfassungen zu sehen, wie viele Mitglieder jeden Artikel geliked und kommentiert haben.

Auf den Artikelseiten wird unter dem Titel angezeigt, wie viele Likes und Kommentare ein Artikel erhalten hat; die vollständigen Kommentare erscheinen unter dem Bild. Ein eingeloggtes Mitglied sieht außerdem:

- das Herz-Symbol mit einem Link, um den Artikel zu liken/zu disliken,
- ein Formular zum Verfassen eines Kommentars zum Artikel (andernfalls wird es aufgefordert, sich einzuloggen, um zu kommentieren).

Quelle: CreativeFolk



Sobald Sie wissen, was die neuen Funktionen den Benutzern ermöglichen,

1. finden Sie heraus, welche Daten in der Datenbank gespeichert werden sollen.
2. schreiben oder aktualisieren Sie die Methoden, mit denen Daten aus der Datenbank abgerufen oder dort gespeichert werden sollen.

3. Erstellen oder aktualisieren Sie die PHP-Seiten, damit sie bei Bedarf die neuen Aufgaben ausführen können, und stellen Sie sicher, dass die Templates die benötigten Daten erhalten.

4. Erstellen oder aktualisieren Sie die Templates, damit die Besucher mit diesen den Funktionen interagieren können.

FESTLEGEN, WELCHE DATEN WIE GESPEICHERT WERDEN SOLLEN

Entscheiden Sie, welche Daten die Nutzer sehen müssen und ob die Datenbank neue Daten speichern muss.

Um Likes anzuzeigen, muss die Datenbank die folgenden Daten speichern:

- Nutzer, dem der Artikel gefällt (bereits in der member-Tabelle)
- Artikel, der ihm gefällt (neue Daten)

Um Kommentare anzuzeigen, muss die Datenbank folgende Daten speichern:

- Nutzer, der den Kommentar abgegeben hat (in der Member-Tabelle)
- Kommentar (neue Daten)
- Datum und Uhrzeit

Als Nächstes müssen Sie entscheiden, wie die neuen Daten in der Datenbank gespeichert werden sollen.

A. Wenn es sich um zusätzliche Daten über bereits in einer bestehenden Tabelle vorhandene Daten handelt (z.B. ein Artikel oder ein Mitglied), fügen Sie sie zu dieser Tabelle hinzu.

B. Wenn es sich um ein völlig neues Konzept oder Objekt handelt, erstellen Sie eine neue Tabelle. Zum Beispiel werden Kommentare in einer neuen Kommentartabelle gespeichert.

C. Wenn eine Beziehung zwischen bereits in der Datenbank vorhandenen Konzepten beschrieben wird, verwenden Sie eine Verknüpfungstabelle. Bei der Implementierung von Likes verfügt die Datenbank bereits über Daten zu Mitgliedern und Artikeln, sodass eine Verknüpfungstabelle die IDs der Mitglieder und die IDs der Artikel, die sie liken, speichert.

KLASSEN UND METHODEN ZUM ABRUFEN UND SPEICHERN VON DATEN ERSTELLEN

Wenn Sie wissen, welche Daten in der Datenbank gespeichert werden müssen, schreiben Sie alle zum Abrufen, Erstellen, Aktualisieren und Löschen dieser Daten erforderlichen Klassen und Methoden. Zwei neue Klassen implementieren Likes und Kommentare:

- Die Like-Klasse sammelt Likes, fügt sie hinzu und löscht sie.
- Die Comment-Klasse ruft die Kommentare der Benutzer ab und fügt sie hinzu.

Die bestehende Klasse Article wird ebenfalls aktualisiert, sodass die Methoden get() und getAll() die Gesamtzahl der Likes und Kommentare jedes Artikels zurückgeben.

PHP-SEITEN AKTUALISIEREN

Als Nächstes müssen Sie herausfinden, ob Sie bestehende PHP-Dateien aktualisieren oder neue erstellen müssen, um die Funktionen zu implementieren.

Zum Beispiel werden in einer neuen Datei Daten gespeichert, wenn ein Besucher einen Artikel liked oder disliked.

TEMPLATE-DATEIEN

Schließlich können die Templates, die den an den Browser zurückgesandten HTML-Code generieren, aktualisiert werden.

Das Template article-summaries.html zeigt, wie viele Likes und Kommentare ein Artikel erhalten hat.

Like-KLASSE

METHODE	BESCHREIBUNG
get()	Prüfen, ob der Artikel geliked wurde
create()	Like in die Datenbank einfügen
delete()	Like aus der Datenbank entfernen

Comment-KLASSE

METHODE	BESCHREIBUNG
getAll()	Kommentare für Artikel abrufen
create()	Kommentar in Datenbank einfügen

Außerdem prüft die bestehende article.php-Seite, ob

- ein Benutzer eingeloggt ist und, falls ja, ob ihm der Artikel gefällt,
- ein Kommentar abgegeben wurde. Wenn ja, wird der Kommentar validiert und kann dann in der Datenbank gespeichert werden.

Das Template article.html zeigt die Gesamtzahl der Likes und Kommentare sowie die vollständigen Kommentare an.

Wenn der Nutzer angemeldet ist, wird ein Link hinzugefügt, um den Artikel zu liken oder zu dislike, und es wird ein Formular angezeigt, über das der Nutzer einen Kommentar abgeben kann.

KOMMENTARE SPEICHERN

Eine neue Tabelle mit der Bezeichnung comment enthält die einzelnen Kommentare und die Kennung der Mitglieder, die diese abgegeben haben.

In der neuen Kommentartabelle (siehe unten) ...

- wird id mit der automatischen Auto-Inkrementierungsfunktion von MySQL erstellt,
- ist comment ein Kommentar zu einem Artikel,
- ist posted das Datum und die Uhrzeit, zu der der Kommentar gespeichert wurde (von der Datenbank in die Tabelle eingefügt),
- ist article_id die ID des Artikels, dem der Kommentar gilt, und
- ist member_id die ID des Mitglieds, das den Kommentar geschrieben hat.

Die Spalten article_id und member_id verwenden Fremdschlüssel-Bedingungen (siehe S. 431), um sicherzustellen, dass sie eine gültige Artikel- und Mitgliedskennung enthalten.

HINWEIS: Erstellen Sie immer eine Sicherungskopie der Datenbank, bevor Sie sie ändern (siehe S. 427). Das ist wichtig, da das Hinzufügen einer neuen Funktion versehentlich Daten überschreiben oder löschen könnte.

Kommentar				
id	comment	posted	article_id	member_id
1	Love this, totally makes me want to...	2019-03-14 17:45:13	24	1
2	I bought one of these guides for NYC...	2019-03-14 17:45:15	24	0
3	Another great piece of work Ivy,...	2019-03-14 17:53:52	3	4

Auf der rechten Seite sehen Sie zwei SQL-Abfragen, mit denen die Gesamtzahl der Kommentare und Likes eines Artikels gezählt wird.

Diese beiden Abfragen werden bei der Anzeige von Artikelzusammenfassungen und einzelnen Artikeln verwendet.

Die neue Tabelle zum Speichern von Likes wird auf der rechten Seite angezeigt.

GESAMTZAHL DER KOMMENTARE:

```
SELECT COUNT(id)
  FROM comments
 WHERE comments.article_id = article.id
```

GESAMTZAHL DER LIKES:

```
SELECT COUNT(article_id)
  FROM likes
 WHERE likes.article_id = article.id
```

LIKES SPEICHERN

Die Datenbank enthält bereits Tabellen für Artikel und Mitglieder. Die neue Tabelle `likes` zeichnet jeden Artikel auf, der einem Mitglied gefällt.

Um alle Artikel zu erfassen, die den einzelnen Mitgliedern gefallen, muss die Datenbank lediglich eine Beziehung zwischen einem einzelnen Mitglied und den jeweiligen Artikeln speichern (da sie bereits Daten über die Mitglieder und Artikel enthält).

Die Beziehung wird durch eine Verknüpfungstabelle beschrieben (sie verknüpft Daten in zwei Tabellen). Ihre Spalten sind:

- `article_id`: die ID des Artikels, der dem Mitglied gefällt
- `member_id`: die Kennung des Mitglieds, dem der Artikel gefällt

Die Verknüpfungstabelle heißt `likes` und ist unten abgebildet.

(Sie verwendet den Plural `likes` und nicht `like`, weil es in SQL das Schlüsselwort `LIKE` gibt, siehe S. 404.)

Die Spalten `article_id` und `member_id` verwenden Fremdschlüssel-Bedingungen (siehe S. 431), um sicherzustellen, dass eine gültige Artikel- und Mitgliedskennung enthalten ist.

Ein Mitglied sollte jeden Artikel nur einmal liken, daher wird ein **zusammengesetzter Primärschlüssel** mit phpMyAdmin hinzugefügt. Er verhindert, dass in einer Zeilen dieselbe Kombination von `article_id` und `member_id` in einer anderen Zeile der Tabelle gespeichert wird – ganz ähnlich wie in der Member-Tabelle zwei Mitglieder nicht dieselbe E-Mail-Adresse haben dürfen.

Mehr über das Erstellen eines zusammengesetzten Primärschlüssels erfahren Sie unter <http://notes.re/php/composite-key>.

likes		member						
article_id	member_id							
1	1							
2	1							
1	2							

article									
id	title	summary	content	created	category_id	member_id	image_id	published	seo_title
1	PS Poster	Poster	Parts...	2019	2	2	1	1	ps-poster
2	Systemic	Leaflet	Design...	2019	2	1	2	1	systemic
3	AQ Website	New site A	new...	2019	1	1	3	1	aq-web

ZUSAMMENFASSUNGEN MIT LIKES UND KOMMENTAREN ANZEIGEN

Die SQL-Abfragen zur Erfassung der Artikeldaten werden um die Erfassung der Gesamtzahl der Likes und Kommentare für jeden Artikel erweitert. Zur Berechnung dieser Gesamtzahlen werden der Hauptabfrage zwei Unterabfragen hinzugefügt.

Die Methode `getAll()` der Klasse `Article` ruft zusammenfassende Daten über eine Artikelgruppe mithilfe einer SQL-Abfrage ab. Diese Methode wird von der home-, category-, member- und search-Seite verwendet.

Um die Gesamtzahl der Likes und Kommentare für jeden Artikel zu erfassen, werden der bestehenden SQL-Abfrage zwei **Unterabfragen** hinzugefügt.

Unterabfragen sind zusätzliche Abfragen, die innerhalb einer anderen Abfrage ausgeführt werden. Die Änderungen an der ursprünglichen Abfrage sind auf der rechten Seite hervorgehoben.

Jedes Mal, wenn die Hauptabfrage eine Artikelzusammenfassung auswählt, um sie der Ergebnismenge hinzuzufügen, werden diese Unterabfragen ausgeführt:

- Die erste zählt die Anzahl der Likes für diesen Artikel.
- Die zweite zählt die Anzahl der Kommentare für den Artikel.

Die Unterabfragen haben dieselbe Syntax wie andere SQL-Abfragen, werden aber in Klammern gesetzt.

Jede Abfrage gibt einen einzelnen Wert zurück, und nach den Klammern definiert ein Alias einen Namen für die Spalte in der Ergebnismenge mit dem Ergebnis dieser Abfrage.

Dieselben beiden Unterabfragen wurden der `get()`-Methode der Klasse `Article` hinzugefügt; Sie finden sie im Code-Download.

HINWEIS: Um diesen Code auf der Seite unterzubringen, wird das Array `$arguments` mit einer Kurzvariante erstellt, mit der Sie zwei Schlüsseln in einer Anweisung denselben Wert zuweisen können.

1. Mit der ersten Unterabfrage wird die Anzahl der Likes aus der Tabelle `likes` abgerufen. Da diese Unterabfragen zusätzliche Daten über jeden Artikel abrufen, werden sie nach dem `SELECT`-Befehl platziert.

Nach den schließenden Klammern gibt ein Alias an, dass die Ergebnismenge die Anzahl der Likes in der Spalte `likes` speichern soll.

2. Die zweite Unterabfrage ruft die Anzahl der Kommentare für einen Artikel ab. Sie ist wie die erste Unterabfrage in Klammern gesetzt.

Der Alias gibt an, dass die Anzahl der Kommentare für den Artikel zu der Spalte `comments` hinzugefügt wird.

3. Das Template `article-summaries.html`, das für die Anzeige von Artikelzusammenfassungen auf den home-, category-, member- und search-Seiten verwendet wird, wird aktualisiert, um die Anzahl der Likes und Kommentare anzulegen. Diese neuen Daten werden vor dem Artikeltitel angezeigt.

Zwar enthalten die PHP-Dateien in `src/pages` für die Home-, Kategorie-, Mitglieder- und Such-Seiten alle Artikelzusammenfassungen, müssen aber nicht aktualisiert werden, da der SQL-Code in der Klasse entsprechend aktualisiert wurde. Die Daten werden dann zu dem Array hinzugefügt, das die Ergebnismenge repräsentiert, und dieses Array wird bereits an das Twig-Templates übergeben.

```

public function getAll($published = true, $category = null, $member = null,
                      $limit = 1000): array
{
    $arguments['category'] = $arguments['category1'] = $category; // Kategorie-ID
    $arguments['member'] = $arguments['member1'] = $member; // Autor-ID
    $arguments['limit'] = $limit; // Max. Artikel
    $sql = "SELECT a.id, a.title, a.summary, a.created, a.category_id,
              a.member_id, a.published, a.seo_title,
              c.name AS category,
              c.seo_name AS seo_category,
              m.forename, m.surname,
              CONCAT(m.forename, ' ', m.surname) AS author,
              i.file AS image_file,
              i.alt AS image_alt,
              (SELECT COUNT(article_id)
               FROM likes
               WHERE likes.article_id = a.id) AS likes,
              (SELECT COUNT(article_id)
               FROM comment
               WHERE comment.article_id = a.id) AS comments
              FROM article AS a
              JOIN category AS c ON a.category_id = c.id
              JOIN member AS m ON a.member_id = m.id
              LEFT JOIN image AS i ON a.image_id = i.id
              WHERE (a.category_id = :category OR :category1 is null)
                    AND (a.member_id = :member OR :member1 is null) "; // SQL
    if ($published == true) {
        $sql .= "AND a.published = 1 "; // Nur veröffentlichte Artikel
    }
    $sql .= "ORDER BY a.id DESC
              LIMIT :limit;"; // Weitere Klauseln
    return $this->db->runSQL($sql, $arguments)->fetchAll(); // Daten zurückgeben
}

```

```

③<div class="social">
<div class="like-count"><span class="icon-heart-empty"></span> {{ article.likes }}</div>
<div class="comment-count"><span class="icon-comment"></span> {{ article.comments }}</div>
</div>
<h2>{{ article.title }}</h2>

```

LIKES HINZUFÜGEN UND ENTFERNEN

Nach der Anmeldung wird das Like-Symbol auf der Artikelseite zu einem Link. Die Datei, zu der der Link führt, prüft, ob der Nutzer den Artikel bereits geliked hat; wenn nicht, wird ein Like hinzugefügt. Wenn ja, wird es entfernt. Zu diesem Zweck werden Methoden der Like-Klasse aufgerufen.

Wenn der Benutzer eingeloggt ist, wird das Herzsymbol auf der Artikelseite in einem Link platziert:

```
<a href="/like/24"> ... </a>
```

Die URL für den Link beginnt mit like, dann folgt die Artikel-ID. Wenn der Link angeklickt wird, wird in index.php die neue Datei like.php eingefügt.

1. In dieser prüft die Bedingung einer if-Anweisung, ob

- es *keine* Artikel-ID gibt oder
- der Benutzer nicht eingeloggt ist (wenn nicht, hat die Eigenschaft id des Session-Objekts den Wert 0).

2. Wenn einer der beiden Fälle zutrifft, sollte der Besucher die Seite nicht nutzen, und er wird zu page not found weitergeleitet.

3. Sonst prüft die get()-Methode eines neuen Like-Objekts, ob dieses Mitglied den Artikel geliked hat. Sie benötigt dazu die Artikel- und Mitglieder-IDs, die als indiziertes Array an die Methode übergeben werden. Das Ergebnis (0 für nein oder 1 für ja) wird in \$liked gespeichert.

4. Wenn dieser Benutzer den Artikel bereits geliked hat, wird die delete()-Methode des Like-Objekts aufgerufen, um den Eintrag aus der likes-Tabelle in der Datenbank zu entfernen.

5. Andernfalls hat er ihn noch nicht geliked, und die create()-Methode der Like-Klasse fügt ein Like hinzu.

6. Dann wird der Benutzer zur Artikelseite zurückgeschickt.

Die neue Like-Klasse aktualisiert die Datenbank, wenn einem Mitglied einen Artikel geliked oder disliked hat. Sie hat drei Methoden:

- get() prüft, ob einem Nutzer der Artikel gefällt
- create() fügt ein Like zur Datenbank hinzu
- delete() löscht ein Like aus der Datenbank

Jede Methode benötigt zwei Daten, die als indiziertes Array an die Methoden übergeben werden:

- die ID des Artikels
- die ID des Mitglieds

Die Like-Klasse ähnelt den Article-, Category- und Member-Klassen. Sie wird mit einer neuen getLike()-Methode des CMS-Objekts erstellt und speichert den Ort eines Datenbankobjekts in der Eigenschaft \$db.

7. Die get()-Methode verwendet die SQL-Methode COUNT(), um zu prüfen, wie viele Zeilen der likes-Tabelle die angegebene Artikel- und Mitglieds-ID haben. Diese Zahl wird dann von der Methode zurückgegeben.

Da diese Tabelle einen zusammengesetzten Primärschlüssel verwendet, gibt die Methode immer nur 1 zurück, um anzuseigen, dass der Benutzer den Artikel geliked hat, oder 0, wenn der Benutzer ihn nicht geliked hat.

8. Die create()-Methode fügt der likes-Tabelle eine neue Zeile mit der Artikel- und der Mitglieds-ID hinzu.

9. Die delete()-Methode entfernt die Zeile aus der likes-Tabelle mit der angegebenen Artikel- und Mitglieds-ID.

```

<?php
① declare(strict_types = 1);                                // Strikte Typisierung
② if (!id or $session->id == 0) {                           // Wenn keine Valide ID
    include APP_ROOT . '/src/pages/page-not-found.php';   // Page not found
}
③ $liked = $cms->getLike()->get([$id, $session->id]);    // Hat Mitglied geliked
④ if ($liked) {                                             // Wenn bereits geliked
    $cms->getLike()->delete([$id, $session->id]);        // Like entfernen
}
⑤ } else {                                                 // Sonst
    $cms->getLike()->create([$id, $session->id]);        // Like hinzufügen
}
⑥ redirect('article/' . $id . '/' . $parts[2] . '/');      // Weiterleitung Artikelseite

```

```

...
public function get(array $like): bool
{
    $sql = "SELECT COUNT(*)
            FROM likes
            WHERE article_id = :id
                  AND member_id = :member_id;";           // SQL
    return $this->db->runSQL($sql, $like)->fetchColumn(); // Ausführen, 1 oder 0
}

public function create(array $like): bool
{
    $sql = "INSERT INTO likes (article_id, member_id)
                  VALUES (:article_id, :member_id);";       // SQL
    $this->db->runSQL($sql, $like);                   // SQL ausführen
    return true;                                       // true zurückgeben
}

public function delete(array $like): bool
{
    $sql = "DELETE FROM likes
                  WHERE article_id = :article_id
                  AND member_id = :member_id;";             // SQL
    $this->db->runSQL($sql, $like);                 // SQL ausführen
    return true;                                       // true zurückgeben
}

```

KOMMENTARE ZU ARTIKELN HINZUFÜGEN

Wenn ein Benutzer angemeldet ist, wird das Formular zum Einreichen eines Kommentars unter dem Bild und allen vorhandenen Kommentaren angezeigt. Die neue Klasse `Comment` verfügt über Methoden zum Abrufen von Kommentaren aus der Kommentartabelle der Datenbank beziehungsweise zum Hinzufügen von Kommentaren zu dieser Tabelle.

Die neue Klasse `Comment` hat zwei Methoden:

- `getAll()` liefert alle Kommentare zu einem Artikel.
- `create()` fügt einen Kommentar zur Kommentartabelle hinzu.
- `getAll()` ruft alle Kommentare zu einem Artikel ab, zusammen mit dem Namen und dem Profilbild des Mitglieds, das den jeweiligen Kommentar abgegeben hat.

Die Methode hat ein Argument: die ID des Artikels, für den sie die Kommentare abrufen soll.

Um den Namen und das Bild des kommentierenden Mitglieds abzurufen, verknüpft der SQL-Code

- die Spalte `member_id` in der `comment`-Tabelle
 - die Spalte `id` in der `member`-Tabelle.
2. Die Methode `create()` fügt einen neuen Kommentar in die `comment`-Tabelle in der Datenbank ein. Sie benötigt drei Daten, die als indiziertes Array an die Methode übergeben werden:

- Kommentar
- Artikel-ID
- ID des Mitglieds, das den Kommentar verfasst hat

Die Auto-Inkrementierungsfunktion der Datenbank erstellt die ID in der ersten Spalte der `comment`-Tabelle. Die Datenbank fügt auch das Datum und die Uhrzeit des Kommentars in die Spalte `posted` der Tabelle ein.

Die Seite `article.php` wird aktualisiert, damit die Kommentare für den aktuellen Artikel angezeigt und alle neuen Kommentare gespeichert werden.

3. Eine `if`-Anweisung prüft, ob die Anfrage eine POST-Anfrage war, was bedeutet, dass das Kommentarformular abgeschickt wurde.
4. Wenn ja, wird der Kommentartext abgerufen.
5. Ein neues `HTMLPurifier`-Objekt wird erstellt, um unerwünschtes Markup aus dem Kommentar zu entfernen.
6. Die Konfigurationsoptionen werden so eingestellt, dass die Elemente `
`, ``, `<i>` und `<a>` zulässig sind.
7. Mit der Methode `purify()` werden alle anderen HTML-Tags aus dem Kommentar entfernt.
8. Wenn der Kommentar zwischen 1 und 2000 Zeichen lang ist, wird ein leerer String in `$error` gespeichert. Andernfalls speichert `$error` eine Fehlermeldung.
9. Wenn kein Fehler auftritt, werden der Kommentar, die Artikel-ID und die ID des Mitglieds in dem Array `$arguments` gespeichert.
10. Dann wird die `create()`-Methode des `Comment`-Objekts aufgerufen, um den Kommentar in der Datenbank zu speichern.
11. Die Artikelseite wird neu geladen, um den Kommentar anzuzeigen.
12. Die `getAll()`-Methode des `Comment`-Objekts ruft alle Kommentare für diesen Artikel ab. Sie werden dem Array `$data` zur Verwendung im Twig-Template hinzugefügt.

```

public function getAll(int $id): array
{
    $sql = "SELECT c.id, c.comment, c.posted,
        CONCAT(m.forename, ' ', m.surname) AS author, m.picture
    FROM comment AS c
    JOIN member AS m ON c.member_id = m.id
    WHERE c.article_id = :id;"; // SQL-Statement
    return $this->db->runSQL($sql, ['id' => $id])->fetchAll(); // Abfrage
}

public function create(array $comment): bool
{
    $sql = "INSERT INTO comment (comment, article_id, member_id)
        VALUES (:comment, :article_id, :member_id);"; // SQL-Statement
    $this->db->runSQL($sql, $comment); // Abfrage
    return true;
}

```

```

<?php ...
③ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Formular abgeschickt
④     $comment = $_POST['comment']; // Kommentar
⑤     $purifier = new HTMLPurifier(); // HTMLPurifier erstellen
⑥     $purifier->config->set('HTML.Allowed', 'br,b,i,a[href]');// Zulässige Tags
⑦     $comment = $purifier->purify($comment); // Kommentar bereinigen
⑧     $error = Validate::isText($comment, 1, 2000)
        ? '' : 'Your comment must be between 1 and 2000 characters.
        It can contain <b>, <i>, <a>, and <br> tags.'; // Kommentar validieren
⑨     if ($error == '') { // Speichern, wenn kein Fehler
        $arguments = [$comment, $article['id'], $cms->getSession()->id,]; // Argumente
        $cms->getComment()->create($arguments); // Kommentar erstellen
        redirect($path); // Seite neu laden
    }
}
$data['navigation'] = $cms->getCategory()->getAll(); // Kategorien
$data['article'] = $article; // Artikel
$data['section'] = $article['category_id']; // Aktuelle Kategorie
$data['comments'] = $cms->getComment()->getAll($id); // Kommentare
if ($cms->getSession()->id > 0) { // Wenn Nutzer eingeloggt
    $data['liked'] = $cms->getLike()->get([$id, $cms->getSession()->id]); // Like?
    $data['error'] = $error ?? null; // Fehler Kommentar
}

```

TEMPLATE FÜR DIE ARTIKELSEITE AKTUALISIEREN

Das Template `article.html` muss aktualisiert werden, um die neuen Daten anzuzeigen, die die Seite `article.php` abgerufen hat. Wenn ein Mitglied angemeldet ist, enthält sie auch einen Link, um den Artikel zu liken, und ein Formular, um den Artikel zu kommentieren.

Auf der rechten Seite sehen Sie die Änderungen am Template `article.html`, das zur Anzeige von Artikeln verwendet wird.

- Zunächst geht es um die Option, einen Artikel zu liken.
1. Eine `if`-Anweisung überprüft, ob die `id`-Eigenschaft des `Session`-Objekts den Wert 0 hat, was bedeuten würde, dass der Besucher nicht angemeldet ist.
 2. Wenn der Besucher nicht eingeloggt ist, wird ein Link zur Anmeldeseite mit einem leeren Herzsymbol eingefügt.
 3. Andernfalls ist der Besucher eingeloggt, und es wird ein Link zur Like-Seite erstellt (er enthält die Artikel-ID). Wenn dieser Link angeklickt wird, fügt die Like-Seite entweder ein Like für dieses Mitglied hinzu oder entfernt es.
 4. Eine weitere `if`-Anweisung prüft den Wert der Variable `liked`, um festzustellen, ob der Besucher den Artikel geliked hat.
 5. Wenn ja, wird ein ausgefülltes Herzsymbol angezeigt.
 6. Wenn nicht, erscheint ein unausgefülltes Herzsymbol.
 7. Der `if`-Block aus Schritt 4 wird geschlossen.
 8. Der `if`-Block aus Schritt 1 wird geschlossen.
 9. Die Gesamtzahl der Likes für den Artikel wird neben dem Herzsymbol angezeigt.

Anschließend werden die Kommentare angezeigt.

10. Die Anzahl der Kommentare für den Artikel wird angezeigt.
11. Eine Schleife wird erstellt, um alle im Kommentar-Array (erstellt in Schritt 12 der vorherigen Seite) gespeicherten Kommentare zu durchlaufen.
12. Wenn Kommentare vorhanden sind, wird das Profilbild der Person, die den Kommentar geschrieben hat, angezeigt, mit ihrem Namen als Alt-Text.
13. Der Name des Mitglieds wird neben dem Foto angezeigt.
14. Das Datum und die Uhrzeit des Kommentars werden angezeigt. Mit dem Twig-Filter `date()` wird das Datum so formatiert wie die anderen Daten auf der Website.
15. Der Kommentar wird angezeigt. Twigs `raw`-Filter verhindert, dass Markups maskiert werden, da der Kommentar vor dem Speichern durch `HTMLPurifier` gelaufen ist.
16. Die Schleife wird für jeden Kommentar im Array wiederholt, und Twigs `{% endfor %}`-Tag schließt die Schleife ab.
17. Wenn der Benutzer angemeldet ist, hat die `id`-Eigenschaft des `Session`-Objekts einen Wert größer 0.
18. Ist der Besucher eingeloggt, wird ihm ein Formular angezeigt, mit dem er einen neuen Kommentar abgeben kann.
19. Andernfalls wird dem Besucher eine Nachricht angezeigt, die besagt, dass er sich anmelden muss, um einen Kommentar abzugeben.

```
① ...
② <div class="social">
③   <div class="like-count">
④     {% if session.id == 0 %}
⑤       <a href="{{ doc_root }}login/"><span class="icon-heart-empty"></span></a>
⑥     {% else %}
⑦       <a href="{{ doc_root }}like/{{ article.id }}">
⑧         {% if liked %}
⑨           <span class="icon-heart"></span></a>
⑩         {% else %}
⑪           <span class="icon-heart-empty"></span>
⑫         {% endif %}
⑬         {{ article.likes }}
⑭       </a>
⑮     </div>
⑯     <div class="comment-count">
⑰       <span class="icon-comment"></span> {{ article.comments }}
⑱     </div>
⑲ ...
⑳ <section class="comments">
⑳   <h2>Comments</h2>
⑳   {% for comment in comments %}
⑳     <div class="comment">
⑳       
⑳       <b>{{ comment.author }}</b><br>
⑳       {{ comment.posted|date('H:i a - F d, Y') }}<br>
⑳       <p>{{ comment.comment|raw }}</p>
⑳     </div>
⑳   {% endfor %}
⑳   {% if session.id > 0 %}
⑳     <form action="" method="post">
⑳       <label for="comment">Add comment: </label>
⑳       <textarea name="comment" id="comment" class="form-control"></textarea>
⑳       {% if error == true %}<div class="error">{{ error }}</div>{% endif %}
⑳       <br><input type="submit" value="Save comment" class="btn btn-primary">
⑳     </form>
⑳   {% else %}
⑳     <p>You must <a href="{{ doc_root }}login">log in to make a comment</a>.</p>
⑳   {% endif %}
⑳ </section>
```

ZUSAMMENFASSUNG

WEITERE FUNKTIONEN HINZUFÜGEN

- SEO-freundliche URLs helfen Suchmaschinen bei der Indexierung von Seiten leichter zu lesen und beschreiben die Seite.
- Die Apache-URL-Rewriting-Engine kann alle Anfragen überprüfen und Regeln befolgen, um einige von ihnen an andere Seiten zu senden.
- Wenn Sie eine neue Funktion hinzufügen, sollten Sie vor dem Programmieren genau festlegen, was die Benutzer tun können.
- Überlegen Sie, wie neue Daten gespeichert werden sollen. Neue Konzepte erfordern eine neue Tabelle, zusätzliche Daten zu einem bestehenden Konzept werden in einer vorhandenen Tabelle gespeichert, und Beziehungen zwischen zwei Konzepten können Verknüpfungstabellen verwenden.
- Eine Unterabfrage ist eine Abfrage, die in einer anderen SQL-Abfrage verschachtelt werden kann.
- Testen Sie neue Funktionen auf einem Testserver (nicht auf dem Live-Server).
- Verwenden Sie eine Kopie der Datenbank für den Testserver und erstellen Sie eine Sicherungskopie der Live-Datenbank, bevor Sie neue Funktionen freigeben.

NÄCHSTE SCHRITTE

Herzlichen Glückwunsch! Sie haben das Ende dieses Buchs erreicht. Nachdem Sie die Grundlagen der Erstellung datenbankgestützter Websites mit PHP kennengelernt haben, können Sie nun entscheiden, was Sie als Nächstes lernen möchten, je nachdem, was Sie mit PHP erreichen wollen.

Sie können versuchen, den Code der Beispieldatenbank zu erweitern, um zum Beispiel ...

- ein Mitgliederverzeichnis zu erstellen. Es könnte ähnlich funktionieren wie Artikelseiten, aber stattdessen alle Mitglieder anzeigen.
- eine Seite hinzuzufügen, die anzeigt, welche Artikel jedes einzelne Mitglied geliked hat.
- ein Benachrichtigungssystem für die Benutzer einzurichten. Die Benachrichtigungen könnten wie Kommentare funktionieren, die nur von dem Benutzer gelesen werden können, für den sie bestimmt sind.
- auf allen Artikelseiten eine Paginierung einzufügen (genau wie bei den Suchseiten).

Sie könnten die Codebasis auch verwenden, um einen anderen Website-Typ zu erstellen. Fügen Sie in diesem Fall Datenbanktabellen und Klassen für die Bereiche der Website hinzu. Zum Beispiel könnte eine Website zum Thema Musik verschiedene Künstler und Genres darstellen. Eine Seite zum Thema Gartengestaltung könnte verschiedene Pflanzenarten und deren Pflege vorstellen. Eine Site zum Thema Kochen könnte verschiedene Rezepte und deren Zutaten zeigen.

Sie könnten sich auch damit beschäftigen, wie Sie anderen Programmen (z. B. Apps für mobile Geräte) Zugriff auf die in der Datenbank gespeicherten Daten und deren Aktualisierung gewähren können. Dazu wird eine sogenannte Programmierschnittstelle (**Application Programming Interface, API**) verwendet.

PHP-Programmierer schreiben Websites nicht immer von Grund auf neu; sie bauen sie oft mit Hilfe eines Frameworks oder eines anderen CMS auf. Sie müssen jedoch die Grundlagen der Erstellung datenbankgestützter Websites kennen (wie in diesem Buch gezeigt), um sie anwenden zu können.

FRAMEWORKS

Frameworks stellen den Code bereit, der für die Ausführung gängiger Aufgaben bei der Entwicklung von Websites und Anwendungen mit PHP erforderlich ist. Zu den beliebtesten Frameworks gehören:

- **Symfony** <https://symfony.com>
- **Laravel** <https://laravel.com>

CMS

Content-Management-Systeme nehmen Ihnen die Arbeit ab, Websites von Grund auf neu zu schreiben. Drei beliebte CMS-Anwendungen sind:

- **WordPress** <https://wordpress.org>
- **Drupal** <https://drupal.org>
- **Joomla** <https://joomla.org>

Sie können nicht nur das Aussehen der Website anpassen, sondern auch WordPress-Plugins, Drupal-Module oder Joomla-Erweiterungen schreiben, die neue Funktionen hinzufügen. (Viele Programmierer geben diese Plugins, Module und Erweiterungen an andere Entwickler weiter.)

INDEX

`$_COOKIES` 334-337
`$_FILES` 286, 290
`$_GET` 236-239
`$_POST` 250-253
`$_SERVER` 190-191
`$_SESSION` 338-343, 620-621
`&& || !` 238
`+ - * / % ** ++ --` 50
`< <= > >= <=>` 54-55
`<? 22-23`
`<?= 46`
`== == != 54-55`
`? : 642`
`?? 238, 250`
`__construct()` 160-163
`__FILE__` 525
`{ { } } { % } 585`
`{ } 72-73, 108`

A

Abfragezeichenfolge 182, 236-239
Abhängigkeit 558
Abrufmodus (PDO) 437, 478
Absoluter Pfad 524-525
Administratoren 615, 624
Aktuelles Arbeitsverzeichnis 524-525
Aliase (Namensräume) 564-565
Aliase (SQL) 418
and 57
Anführungszeichen 24
Anmeldesystem (einfach) 344
Anmeldung (fortgeschritten) 614-621
Anonyme Funktionen 553
Antwortcodes (HTTP) 242, 354, 366-367, 378-379
Antwort siehe HTTP-Antwort
Anweisung 18, 23
Apache (Webserver) 20
Arbeitsverzeichnis 524-525
Argumente 114-116, 122, 130-133
Arithmetische Operatoren 49-51
`array()` 38, 40

`array_key_exists()` 218-219
`array_merge()` 220
`array_pop()` 220-221
`array_push()` 220-221
`array_rand()` 220-221
`array_search()` 218-219
`array_shift()` 220
`array_unique()` 220-221
`array_unshift()` 220-221
Arrays 37-45
`arsort()` 222
AS (SQL) 418
Assoziative Arrays 37-39, 42-43
Ausdrücke 30, 48
Äußere Verknüpfung (SQL) 415
Ausnahmen 350, 368-377
auslösen und abfangen 370-373
`try... catch` 370-373
Auswertung (Ausdruck/
Bedingung) 48-59
Authentifizierung (Mitglieder) 614-621
Autoloading 534, 553, 570-571
Automatisches Inkrement 387, 424, 489

B

`basename()` 228, 513
`beginTransaction()` (PDO) 509, 514-515, 518-519
Benannte Parameter und
Argumente 132-133
Benennen 34
Benutzerdefinierte Fehler 354-355
Benutzerkonto (Datenbank) 394-395, 437
Bibliothek 558, 566-567
Bild-Upserts 286-297, 512-513
Bild beschneiden 300-301, 306-307
Bildgröße ändern 298-307
`bin2hex()` 626
`bindParam()` 450
`bindValue()` 450

block (Twig) 577, 588-589
Boolescher Wert 30, 35, 61, 73
Bootstrap-Datei 523, 529, 581
Breakpoints 363
Browserdaten 234-235

C

Bedingte Anweisungen 70-80
Bedingungen in Twig 586
Cache (Twig) 579
`catch` 368-375, 438-439
`ceil()` 216-217
Child-Template 577
`class` 154
CMS (Content-Management-
System) 384
`COALESCE()` (SQL) 420
Code-Block 72-73, 108
`commit()` (PDO) 509, 514,
518-519
Composer 538, 567-571
`CONCAT()` (SQL) 420
`const()` 224-225
`__construct()` 160-163
Constructor property promotion 161
Container-Objekt 539
Cookies 332-337
`$_COOKIES` 334-337
aktualisieren 336-337
Einführung 332-333
erstellen 334-335
`count()` (PHP-Array) 218-219
`COUNT()` (SQL) 408, 472

D

Dateien
`$_FILES` 286, 290
`__FILE__` 525
`basename()` 228
Bilder-Upload 286-297,
512-513
Datei-Upload, Kontrolle 288
`dirname()` 228
`file_exists()` 228
`filesize()` 228-229

- Löschen 228
 Maximale Größe 295-296
`mime_content_type()` 228-229
 Namen duplizieren 294
`pathinfo()` 228-229
`realpath()` 228
`unset()` 228-229
 Validierung 296-297
- Datenbank
 Auto-Inkrement 387, 424, 489
 Backup 427
 Bedingungen 430-431, 491, 500-501, 504-505
 Benutzerkonto 394-395, 437
 Datentypen 388
 Eindeutigkeitsbedingung 430, 491
 Feld 387
 Fremdschlüssel 389, 412
 Fremdschlüsselbedingung 431, 505
 Primärschlüssel 387, 389, 412
 sichern 427
 Transaktionen 508-509, 514-515
 Verbindung 439
 Verknüpfungstabellen 651
 zusammengesetzter Primärschlüssel 651
- Datenbankdaten erstellen *siehe* `INSERT`
 Datenbank erstellen 392
 Datenbank suchen 472-477
 Datentypen 30, 35, 60-61, 388, *siehe auch Array, Boolean, Float, Integer, Object, String*
 Daten bereinigen 235, 246-247, 280-281
 Datum und Zeit 310-328
`date()` 316-317, 585
`date_create_from_format()` 318-319
`DateInterval` 322-323
`DatePeriod` 324-325
`DateTime` 318-319
`DateTimeZone` 326-327
- Datum und Zeit aktualisieren 320-321
 Datum und Zeit festlegen 320-321
 Formate 312-314
`mktime()` 316-317
`strtotime()` 316-317
`time()` 316-317
 Unix-Zeitstempel 315
`declare()` 126
`define()` 224-225, 525, 528-529
 Deklaration einer Variablen 32
 Deklarationen von Argumenttypen 124-127
 Dekrement-Operator 50
`DELETE (SQL)` 428, 488
 Dependency Injection 534, 538-545
 Dependency Manager 558
`DIRECTORY_SEPARATOR` 306
`dirname()` 228
`display_errors` 352-353
 Dokumentenstamm 526-527, 638
 Doppelte Datenbankeinträge
siehe Eindeutigkeitsbedingung
`do while`-Schleife 81, 84-85
 DRY (Don't Repeat Yourself) 170
 DSN (Data Source Name) 436-437, 542-443
 Dynamische Dateinamen 294
 Dynamische vs. statische Websites 4-5, 178
 Standardwerte (Funktionen) 130-131
- E
- `e() -Filter (Twig)` 585
 E-Mail (Senden) 594-601
 E-Mail erstellen 596-599
 E-Mail senden 596-599
`echo` 24-25, 32, 47
 Eigenschaften von Objekten 144, 148-149, 156-157
- Eindeutigkeitsbedingung (Datenbank) 430-431, 491, 500-501, 504-505
 Eindeutigkeitsbeschränkungen 430, 491
 Einfache vs. doppelte Anführungszeichen 24
 Eingebaute Funktionen 136-137, 188, 192-93, 201-230
 Entitäten 235, 244-247
 Entschlüsselung 184
 Erforderliche Daten 234
 Ergebnismenge (SQL) 400
`ErrorDocument` 378
`errorInfo (PDO)` 491
 Erweiterungen 302-303
 Escape-Zeichen 24, 235, 244-247, 585
`explode()` 218
`extends (Twig)` 577, 588-589
- F
- Fehler 189, 194-195, 350-380
 Abschaltfunktion 365, 376-377
 Ausnahmen 368-377
 Benutzerdefinierte Fehlerseiten 354-355
`display_errors` 352-353
 Error-Klasse und Error-Ausnahmen 368-372
`error_log()` 366-367, 372-373, 376-377
`error_reporting` 352-353
 Fatale Fehler 355, 358-359, 365
 Funktionen zur Behandlung von Ausnahmen 371-377
 Funktionen zur Fehlerbehandlung 366-367, 376-377
 Hinweise 355
 Lesefehler 354
`log_errors` 352-353
 nicht schwerwiegende Fehler 355, 360-361
 Parse-Fehler 355-357

- Protokolldateien 352, 364
Stufen 194, 354-355
Warnungen 355, 360-361
Fehlermeldungen (PHP) 189,
194-195, 349-380
Fehlermodus (PDO) 437
Fehlersuche 362-363, 579
Fehlervalidierung *siehe*
Validierung
Feld 387
fetch() (PDO) 443-445
fetchAll() (PDO) 443, 446
finally() 370
Filter 230, 268-273
Bereinigung 280-281
filter_input() 268-269,
272-273, 452
filter_input_array()
268-269, 274-275
filter_var() 276
filter_var_array()
276-277
Flags und Optionen 272-273,
278-79
Überprüfungsfilter 270-271,
278-279
Filter (Twig) 585
float 30, 35, 61
Fluss der Steuerung 68, 484,
496-297
for() 216-217
for-Schleife 81, 86-89, 587
foreach-Schleife 81, 90-93
format() 318-319
Formulare (Daten abrufen von)
248-253
Formularüberprüfung 254-269
Fremdschlüssel-Beschränkungen
431, 505
Fremdschlüssel 389, 412
FROM (SQL) 400, 412-413
Funktionen 104
Argumente 115
Argumenttyp-Deklarationen
124-127
Aufrufen 108-111
- benannte Parameter
(benannte Argumente)
132-133
benutzerdefiniert vs. eingebaut
136
Bezeichnungen 117
Definitionen 108-111
eingebaute 136-137, 188,
192-193, 201-230
Geltungsbereich (lokal vs.
global) 118-121
optionale Parameter und
Standardwerte 130-131
Parameter 114-116
Rückgabetyp-Deklarationen
124-127
Rückgabewerte 112-113,
128-129
strenge Typen 126-127
- G**
- \$_GET 236-239
Ganzzahl (int) 30, 35, 388
Garbage collection 342-343, 531
GD 302-305
Geschweifte Klammern 72-73,
108, 585
Geschäftsvorgänge 484,
508-509, 514-515
Getter 164
Globaler Bereich 118-121
Globaler Namespace 560
globales Schlüsselwort 120
Globale Variablen (Twig) 580
GROUP BY (SQL) 408
Großbuchstaben 204-205
Gültigkeitsbereich (lokal und
global) 118-121
- H**
- Hashes (Passwörter) 608-609,
612-613
header() 226-227
Hinweise (Fehler) 355, 360
Hostname 21
- .htaccess 126, 196, 199,
352-353, 378-379, 638,
640
HTML-Cleaner 558, 572-575,
656-657
htmlspecialchars()
246-247
HTTP-Anfrage und -Antwort
179-183
HTTP-Antwort-Statuscodes 181,
370
HTTP-Kopfzeilen 180-182,
226-227
HTTP-Verweise 190-191
HTTP 179
http_response_code()
242, 370
httpd.conf 196
HTTP GET 180-183, 236-239,
248-253
HTTP POST 180-183, 248-253
HTTPS 184-185
Versteckte Dateien 196
- I**
- if-Anweisung 70-74, 586
if... else-Anweisung 70-71,
73, 75-77, 586
if... elseif-Anweisung
70-71, 78, 586
Imagick und ImageMagick
302-303, 306-307,
514-515
implode() 218-219
in_array() 218, 260-261
include und include_once
94-97
Indexnummern 37, 40-42
Indizierte Arrays 37, 40-42
ini_get() 353
Initialisieren 36, 74
Initialisierung von Variablen 36,
74
Inkrement-Operator 50
Innere Verknüpfung (SQL) 415
INSERT (SQL) 424, 486

Integritätsbeschränkung 431,
505
`is_numeric()` 216-217, 254
`isset()` 238, 262-263
Iteration 68, *siehe auch Schleifen*

J

`JOIN (SQL)` 412-417
`JSON` 226

K

Kennwort-Hashes 607-609,
612-613
Kennwort zurücksetzen 625-631
Kennwortüberprüfung 258-259
Klassen 144, 151-176, 536-555
 Definition 151, 154-155
 Eigenschaften 144, 156-157
 Getter/Setter 164
 Klassendateien 170
 Methoden 144, 158-159
 `private` 164
 `protected` 164
 `public` 164
Kleinbuchstaben 204-205
Kommentare (im Code) 26-27
Kommentare (von Benutzern)
 634, 648-659
Konfigurationsdatei 523, 528,
 595
Konstanten 224-225, 525,
 528-229
Konstruktor 160-163
Kontrollfluss 68, 484, 496-497
Kontrollstrukturen 68-102
`krsort()` 222
`ksort()` 222-223

L

`lastInsertId() (PDO)` 490
`LIKE (SQL)` 404
Like-Feature 634, 648-659
`LIMIT (SQL)` 410, 472
Linke Tabelle 412, 651
`log_errors` 352-353

Logische Operatoren 49, 56-57,
59
`ltrim()` 208-209

M

Magische Methoden 160
MAMP 20
MariaDB 15, 386
`match() expression` 71, 80
`mb_stripos()` 210
`mb_stristr()` 210
`mb_strlen()` 210-211,
 256-259
`mb_strpos()` 210
`mb_stripos()` 210-211
`mb_strripos()` 210-211
`mb_strrstr()` 210
`mb_strtolower()` 210
`mb_strtoupper()` 210
`mb_substr()` 210
Medientyp 290, 295
Mehrbyte-String-Funktionen
 210-211
Mehrdimensionale Arrays
 44-45
Methoden 144, 148-149, 154,
 158-159
Methodenverkettung 457
MIME-Typ *siehe Medientyp*
`mime_content_type()`
 228-229
`mixed (data type)` 35
`mktime()` 316-317
`move_uploaded_file()` 292
`mt_rand()` 216-217
Multi-Byte-Zeichen 187
MySQL 12-13, 15, 382

N

Namensräume (Namespaces)
 558, 560-565
Namespaces importieren 564
`new` 155
`not` 57
Null-Koaleszenz-Operator (??)
 238, 250

`null` 35
`number_format()` 216-217

O

Objekte 144-176
 `__construct()` 160-163
 Constructor Property
 Promotion 161
 Datentyp 150
 Eigenschaften 144, 148-149,
 156-157
 Getter/Setter 164
 Klassen 143-176
 Methoden 144, 148-149,
 158-159
 Referenz 150
 Sichtbarkeitsschlüsselwörter
 (SQL) 410, 472
Operatoren 30, 48-59
 Arithmetik 49, 50-51
 Logisch 56-57
Null-Koaleszenz (??) 238, 250
Raumschiff 55
Ternär 71, 76-77, 238
Vergleich 49, 54-55
Verkettung (String) 49, 52-53
Zeichenkette 49, 52-53
Zuweisung 32-33, 52
`or` 57
`ORDER BY (SQL)` 406

P

Packagist 567-568
Page not Found 242, 378-379,
 452
Paginierung 472-477
Paket 558, 567-571
Parameter (Funktionen) 114-116,
 130-133
Parse-Fehler 355-357
`password_hash()` 609,
 612-613
`password_verify()` 609,
 612-613, 619
`pathinfo()` 228, 294
PDO 382, 434

`beginTransaction()`
 (PDO) 509
`commit() (PDO)` 509
`lastInsertId() (PDO)`
 489
`pdo() (function)` 456
`query()` 443
`rollBack() (PDO)` 509
`PDOException` 438–439, 491
 `errorInfo` 491, 501, 505
`PDOStatement` 434, 443
 `fetch()` 443
 `fetchAll()` 443
 `query()` 443
 `rowCount()` 490
`PDO::ATTR_DEFAULT_FETCH_MODE` 437
`PDO::ATTR_EMULATE_PREPARES` 437
`PDO::ATTR_ERRMODE` 437
`PDO::FETCH_ASSOC` 478
`PDO::FETCH_CLASS` 478
`PDO::FETCH_OBJ` 478
`PDO::PARAM_BOOL` 450
`PDO::PARAM_INT` 450
`PDO::PARAM_STR` 450
Personalisierung 5, 615, 622–623
Pfad 324–325
PHP-Block 22–23
PHP-Interpreter 5–6
PHP-Tags 22–23
PHP-Zeichen 536, 560–561
`php.ini` 196–198, 352, 364
PHP.net 136–137
`phpinfo()` 197
PHPMailer 558, 594–601
`phpMyAdmin` 382, 390–395,
 401
Platzhalter (SQL) 404, 448–449
Portnummern 21, 390, 436
`$_POST` 250–253
`pow()` 216–217
`preg_match()` 214–215,
 258–259
`preg_match_all()` 214–215
`preg_replace()` 214–215,
 294

`preg_split()` 214–215
Primärschlüssel 387, 389, 412
 zusammengesetzter 651
`private` 164
Protokolldateien (Fehler)
 352–353, 364, 366–367
`public` 164
Purifier siehe `HTMLPurifier`

Q

`query() (PDO)` 443

R

Rückgabetyp-Deklarationen
 124–127
Rückgabe (Werte von
 Funktionen) 112–113,
 128–129
`rtrim()` 208–209
`rsort()` 222
`rowCount()` 490
`round()` 216–217
Root-Verzeichnis 524–527
Root-Konto 394
Rollen (Mitgliedschaft) 607, 625
`rollBack() (PDO)` 509, 514–515,
 518–519
Reservierte Zeichen 244–246
require und `require_once` 94–97
Rendervorlage 578
Relativer Pfad 21, 524–525
Relationale Datenbank 13, 382,
 386
Reihenfolge der Ausführung 50
Reguläre Ausdrücke 212–215,
 258–259, 271
Registrierung (Mitglieder)
 610–613
`register_shutdown_function()` 365,
 376–377
Refaktorierung 534, 552
Rechte Tabelle 412
`realpath()` 228–229
RDBMS 386
`raw() -Filter (Twig)` 585

`random_bytes()` 626
`rand()` 216–217

S

`$_SERVER` 190–191
Salt 608
Schleifen 81–93
Schlüssel siehe `Arrays`
Schwerwiegende Fehler
 355–357, 365
`SELECT (SQL)` 400
Semikolon (zum Beenden einer
 Anweisung) 23
SEO-freundliche URLs 634–647
Server-seitige Programmierung 5
`set_error_handler()`
 365–367, 376–377, 529
`set_exception_handler()`
 371, 376–377, 529
`setCookie()` 334–337
`setFetchMode()` 478
Setter 164
Sichere Interaktionen 183
Sichtbarkeit von Eigenschaften
 und Methoden 164
Sitzungen 338–343, 614,
 620–621
`$_SESSION` 338–343,
 620–621
beenden 343
Dauer 342–343
Einführung 338–339
Highjacking 346
`session_destroy()` 340
`session_get_cookie_params()` 340
`session_regenerate_id()` 340
`session_set_cookie_params()` 340
`session_start()` 340
Zugriff auf Sitzungsdaten 341
Skalare Datentypen 35
SMTP-Server 594–595
`sort()` 222
Spaceship-Operator 55

Speichern von Daten in Sitzungen 341
Speicherort für das Hochladen temporärer Dateien 290
`spl_auto_register()` 529, 553, 567
Suchbedingung 402
Suchzeichenfolge (Suchen und Ersetzen) 206–209
SQL 382, 397–432
 Abfrage 398–401
 Anweisung 398
 AS (Alias) 418
 `COALESCE()` 420
 `CONCAT()` 420
 `COUNT()` 408
 `DELETE` 428–429, 488, 505, 518–519
 Eindeutigkeitsbedingung 430
 Einführung 398
 Fremdschlüssel 431
 `FROM` 400
 `GROUP BY` 408
 `INSERT` 424–425, 486, 501
 Integritätsbedingung 430–431, 505
 `JOIN` 412–417
 `LIKE` 404
 `LIMIT` 410
 `OFFSET` 410
 `ORDER BY` 406
 Platzhalter 448–449
 Primärschlüssel 412–414, 424, 426, 428, 431
 `rowCount()` 490
 `SELECT` 400
 `SET` siehe `UPDATE`
 Suche 404–405, 472–477
 Transaktionen 508–509, 514–515
 Unterabfragen 652
 `UPDATE` 426–427, 487, 496–497, 501
 `VALUES` siehe `INSERT`
 `WHERE` 402

`sqrt()` 216–217
SSL 184–185, siehe *HTTPS*
Stammverzeichnis der Anwendung 524–527, 638
Statische Methoden 554–555
Statische Variablen 120
Statische vs. dynamische Websites 4–5
`str_contains()` 206–207
`str_ends_with()` 204–205
`str_ireplace()` 208–209
`str_repeat()` 208–209
`str_replace()` 208–209
`str_starts_with()` 204–205
`str_word_count()` 204–205
Strikte Typen 126–127
String-Funktionen 204–211
 `stripes()` 206–207
 `strpos()` 206–207
 `strripos()` 206–207
 `strrpos()` 206–207
 `strrstr()` 206–207
 `strtol()` 204–205
 `strtolower()` 204–205
 `strtotime()` 316–317
 `strtoupper()` 204–205
String-Operatoren siehe Operatoren
`substr()` 206–207
Superglobale Arrays 179, 188, 190–191, siehe auch `$_COOKIE`, `$_FILES`, `$_GET`, `$_POST`, `$_SERVER`, `$_SESSION`
switch-Anweisung 71, 79
Symbol-Tabellen 530–531

T

Template (Twig) 576–593
Ternary-Operator 71, 76–77, 238
`throw` 369, 373
`Throwable` 369

`time()` 316–317
TinyMCE 572
TLS 184–185, siehe *HTTPS*
Token 625–631
Transaktions-E-Mails 594
Transliteration 644–645
`trim()` 208–209
`try... catch` 370–375, 439, 491, 508–509
Twig 558, 576–593
Typenjoglage 35, 60–61

U

`ucwords()` 204–205
Umleitung 226, 242–243, 495
Undefiniert... (Fehler) 359
Unerwartet... (Fehler) 356–357
Unix-Zeitstempel 315
`unset()` 228–229
Unterabfragen 652
`UPDATE (SQL)` 426, 487
URL-Rewriting siehe SEO-freundliche URLs
UTF-8 187
Übergeordnete Klasse 542
Übergeordnete Vorlage 577
Überprüfung gesendeter Formulare 252–253

V

Validierung 234, 240–241, 254–263, 270–279, 282–283, 498–499, 512–513, 554–555
Bilder 296–297, 512–513
Dateien 294, 296–297, 512–513
E-Mail 271
Filter 270–273, 278–279
Formulare 234, 250–269, 282–283, 498–499
Kennwörter 258–259
Kontrollkästchen 262–263

Optionen (Auswahl/Radio) 260–261
Reguläre Ausdrücke 258–259, 271
Textlänge 256–257
URI 271
Ziffern 254–255, 270
`var_dump()` 192–193, 362
Variablen 8, 11, 30–36
Verbindung zur Datenbank 438–439
Vererbung (Templates) 577, 588–589
Vergleichsoperatoren 49, 54–55
Verkettungsoperator 48–49, 52
Verschlüsselungsverfahren 179, 186–187, 436

Verschlüsselung 184
Verweis (auf Objekt) 150
Verweis (Speicher) 530–531
W
Warnungen (Fehler) 355, 360–361
WHERE (SQL) 402
while-Schleife 81–83
Wortanzahl 204–205
WYSIWYG-Editor 572

X

XAMPP 20
XSS-Angriff 244–247, 336, 572–575, 585

Z

Zahlen *siehe Integer und Float*
Zeichencodierung 186–187, 436
Zeichenkette 30, 35, 52, 61
Zeit und Datum *siehe Datum und Zeit*
Zeitzonen 326–227
Zertifizierungen 185, *siehe HTTPS*
Zufallszahl 216–217
Zusammengesetzte Datentypen 37, 150
Zusammengesetzter Primärschlüssel 651
Zuweisungsoperator 32
Zuweisung von Werten 32
Zähler (Schleifen) 81–82, 84, 86–87

LADEN SIE DEN CODE FÜR DIESES BUCH HERUNTER:

<http://phpandmysql.com>

IMPRESSUM

AUTOR

Jon Duckett

ERGÄNZENDES MATERIAL

Chris Ullman

GESTALTERISCHE LEITUNG

Emme Stone

FACHLEKTORAT

Roman Schevchenko
Art Bergquist
Jack Shepler
Phil DeGeorge

GUTACHTER

Bob Erickson
Chris Dawson
Scott Weaver
Trevor Reynolds

DANKSAGUNG

Jim Minatel
Alcwyn Parker
Daniel Morgan
Richard Eskins

LERNEN SIE

PHP-Code zu lesen und zu schreiben
Daten in einer MySQL-Datenbank
zu speichern
Seiten auf jeden Besucher
zuzuschneiden
ein CMS oder soziales Netzwerk
aufzubauen.

TECHNIKEN

Verwaltung von Inhalten
Registrierung und Mitgliedschaft
Hochladen von Bildern und
Medien, Kommentaren und Likes

ONLINE-UNTERSTÜTZUNG

Code-Beispiele und praktische
Übungen sind online verfügbar
unter: www.phpandmysql.com
Außerdem finden Sie dort
zusätzliche Materialien.

Willkommen bei der schöneren Art, PHP und MySQL zu lernen. PHP ist eine Programmiersprache, mit der viele der weltweit führenden Websites betrieben werden (darunter Facebook und Wikipedia). PHP läuft auf Webservern und ermöglicht es, Webseiten für Besucher individuell anzupassen, indem Inhalte aus einer MySQL-Datenbank abgefragt werden. Die einfachen, visuellen Erklärungen und passenden Code-Beispiele in diesem Buch machen es Ihnen leicht, mit PHP & MySQL Websites zu entwickeln, die es Besuchern ermöglichen, sich als Mitglieder zu registrieren, Artikel zu erstellen und zu bearbeiten, Bilder hochzuladen, Profile zu verwalten, Beiträge zu kommentieren oder zu „liken“ und vieles mehr ...

JON DUCKETT hat als Webdesigner und -entwickler viele Jahre für kleine Startups und große Firmen gearbeitet. Außerdem ist er Autor zahlreicher Bücher zu Webdesign und -programmierung, Usability und Barrierefreiheit.

ISBN 978-3-527-76070-1



9 783527 760701 >

WILEY

Webdesign/Webentwicklung