

# Lösungen zu den Übungen

Dieses Begleitkapitel unterstützt Sie beim Verständnis der Lösungen zu den Übungen im Buch. Darüber hinaus erhalten Sie Hintergrundwissen und weiterführende Tipps. Am besten studieren Sie je Übung zunächst den kommentierten Lösungs-Quelltext und lesen dann die ergänzenden Beschreibungen aus diesem Kapitel.

Das Verzeichnis `php-mysql-schnelleinstieg/` enthält alle begleitenden Quelltexte zum Buch, geordnet nach Kapiteln. Die Lösungs-Quelltexte zu den Übungen finden Sie in jedem Kapitelverzeichnis im Unterverzeichnis `_Übungen`. Der Lösungs-Quelltext zu Übung 1 aus Kapitel 6 befindet sich beispielsweise unter:

```
php-mysql-schnelleinstieg/kapitel-06/_Übungen/Übung-1
```

Beachten Sie auch die allgemeinen Hinweise zur Nutzung der Quelltexte in `php-mysql-schnelleinstieg/README.md`.

Lösungen für Kommandozeilen-Skripte können Sie direkt mit dem `php`-Befehl ausführen:

```
> cd kapitel-01/_Übungen/Übung-1/  
> php two-dices.php  
Sie haben eine 2 und eine 5 gewürfelt.
```

Für Lösungen zur Erzeugung dynamischer Webseiten starten Sie zunächst einen Webserver mit der passenden Document Root, etwa mit der Option `-t` des `php`-Befehls:

```
> php -S localhost:8000 -t kapitel-06/_Übungen/Übung-1/
```

... oder direkt im passenden Verzeichnis:

```
> cd kapitel-06/_Übungen/Übung-1/
```

```
> php -S localhost:8000
```

Ein Abruf der Lösung erfolgt dann über den Browser, z.B. für den im Beispiel gestarteten Webserver für die Lösung zu Übung 1 aus Kapitel 6:

```
http://localhost:8000/list-http-request-headers.php
```

Viel Spaß und Erfolg beim Programmieren!

Philipp Rieber

@bicpi

## 1.1 Kapitel 1

### Übung 1

Für eine unabhängige, zweite Zufallszahl zwischen eins und sechs ruft das Lösungsskript die Funktion `random_int(1, 6)` ein zweites Mal auf.

### Übung 2

Die `date()`-Funktion liefert das aktuelle Datum im gewünschten Format, welches in den Klammern in Anführungszeichen gesetzt angegeben ist. Ein vorangestelltes `echo` sorgt für die Ausgabe des Ergebnisses:

```
echo date('Gewünschtes Format');
```

Das gewünschte Format drücken Sie mit Buchstaben aus, z.B. `Y` für eine vierstellige Jahreszahl oder `H` für die Stunde im 24-Stunden-Format.

Die PHP-Dokumentation beschreibt die `date()`-Funktion unter <https://www.php.net/manual/de/function.date>. Die Beschreibung des Parameters `$format` enthält einen Link zur Übersicht aller Formatzeichen: <https://www.php.net/manual/de/datetime.format>.

## 1.2 Kapitel 2

### Übung 1

<code>\$book</code>	Ungültig; beginnt nicht mit einem Dollarzeichen.
<code>\$eBook</code>	Gültig.
<code>\$_1</code>	Gültig.
<code>\$1_</code>	Ungültig; eine Zahl zu Beginn des Bezeichners ist nicht zulässig.
<code>\$Irobot</code>	Gültig.
<code>\$te\$t</code>	Ungültig; ein Dollarzeichen im Bezeichners ist nicht zulässig.
<code>\$bücher</code>	Gültig.
<code>VERSION</code>	Ungültig; PHP würde dies als Namen einer Konstante interpretieren.

### Übung 2

Die Eingaben, die Sie von der Funktion `readline()` für Höhe, Breite und Tiefe erhalten, sind immer Strings, d.h. Text. Um im Anschluss sicher eine sinnvolle Rechnung (mit Zahlen) durchzuführen, erfolgt in der Lösung eine sofortige Umwandlung in Ganzzahlen durch ein Type Casting mit dem `(int)`-Operator. Bei der Eingabe von Buchstaben für einen der Werte ergibt die Umwandlung die Zahl null.

Die Formel zur Berechnung des Quader-Volumens lautet:  $\text{Höhe} \times \text{Breite} \times \text{Tiefe}$ .

Die Multiplikation ergibt Kubikzentimeter ( $\text{cm}^3$ ). Ein Kubikzentimeter ist ein tausendstel Liter. Die Division der Kubikzentimeter führt daher zum Ergebnis in Litern.

### Übung 3

Gerade und ungerade Zahlen lassen sich anhand des Rests einer Division durch die Zahl zwei unterscheiden. Der Rest kann nur null oder eins sein. Beim Rest null handelt es sich um eine gerade, ansonsten um eine ungerade Zahl. Den Rest einer Division bestimmt der Modulo-Operator `%`.

Der Divisionsrest 0 oder 1 kann praktischerweise direkt für eine Fallunterscheidung in der Bedingung eines ternären Operators oder einer `if`-Bedingung genutzt werden. PHP wertet durch seine automatischen Typumwandlungen die Zahl 0 als `false` und 1 als `true` aus.

### Übung 4

Die Funktion `var_dump()` ist immer nützlich, wenn es darum geht, den Inhalt einer Variablen lesbar darzustellen. Der Aufruf ...

```
var_dump($directoryTree);
```

... kann Ihnen daher in der Übung helfen, den aktuellen Zustand des Arrays `$directoryTree` zu überprüfen.

Ein mehrdimensionales Array kann den Verzeichnisbaum gut abbilden. Jeder Verzeichnistrenner (in der Übung der Schrägstrich `/`) entspricht einer weiteren Dimension im Array. Die Verschachtelung kann beliebig tief gehen. Ein String-Wert im Array stellt eine Datei im Verzeichnis dar. Ein Array eröffnet ein neues Unterverzeichnis mit dem Array-Schlüssel als Verzeichnisname.

### Übung 5

Ausführliche Informationen zum »Type Juggling« von PHP finden Sie unter [www.php.net/manual/language.types.type-juggling](http://www.php.net/manual/language.types.type-juggling). Auf den dort verlinkten Unterseiten zum Casting-Verhalten der einzelnen Datentypen finden Sie unter anderem Informationen, wie sich der Wert `null` in einer Typumwandlung verhält.

## 1.3 Kapitel 3

### Übung 1

<code>0 &gt;= 2</code>	<code>false</code>
<code>'drei' == 'drei'</code>	<code>true</code>
<code>'3' === 3</code>	<code>false</code>
<code>[] === []</code>	<code>true</code>
<code>true    false</code>	<code>true</code>
<code>true &amp;&amp; false</code>	<code>false</code>
<code>!false &amp;&amp; true</code>	<code>true</code>
<code>7 &gt; 4 &amp;&amp; 7 &lt; 6</code>	<code>false</code>
<code>7 &gt; 4 &amp;&amp; 7 &lt; 6    7 &lt; 9</code>	<code>true</code>
<code>' ' == false</code>	<code>true</code>
<code>' ' === false</code>	<code>false</code>

### Übung 2

Bedenken Sie, dass die Funktionen `readline()` und `date()` Strings zurückliefern. Da im Kontext des Programms nur Ganzzahlen sinnvoll sind, erlaubt es das Type Casting mit dem `(int)`-Operator, die zurückgelieferten Werte der Funktionen sicher und ohne Type Juggling zu vergleichen.

Das Lösungsskript bestimmt mit Hilfe der `date()`-Funktion das aktuelle Jahr zum Ausführungszeitpunkt. Daraus berechnet sich dynamisch der für das eingegebene Geburtsjahr erlaubte Wertebereich zwischen dem aktuellen und dem Jahr vor 125 Jahren.

### Übung 3

Die Lösung verwendet die Variable `$sum`, um darin nach und nach die Zahlen aus dem Array zum Ergebnis aufzusummieren. Initial startet sie mit dem Wert 0. Bei jeder Iteration über das Array mit den Zahlen addiert der

kombinierte Zuweisungsoperator die Zahl der aktuellen Iteration zur Ergebnisvariable.

### Übung 4

Fünf Zeilen entstehen durch eine `for`-Schleife mit fünf Iterationen, wobei am Ende jeder Iteration ein Zeilenumbruch ausgegeben wird. Die dynamische Anzahl an Spalten mit Ausgabe der Sterne entsteht durch eine weitere, verschachtelte Schleife innerhalb der ersten Schleife. Die Abbruchbedingung der inneren Schleife richtet sich dynamisch nach der aktuellen Zeilennummer. So kommt in jeder Zeile eine Spalte mit einem weiteren Stern hinzu.

### Übung 5

Die `while`-Schleife eignet sich für eine zuvor unbekannte Anzahl an Iterationen. Die Schleife in der Lösung läuft so lange, bis das Skript sechs gültige Lottozahlen vom Benutzer abgefragt hat. Ist eine Zahl ungültig, d.h. außerhalb des Bereichs 1 bis 49 oder bereits im Lottotipp enthalten, springt das Skript wieder zum Beginn der Iteration, ohne die Zahl in den Tipp aufzunehmen. So gelangen nur gültige Zahlen in den Lottotipp, bis sechs Zahlen gesammelt wurden, egal wie oft der Benutzer eine ungültige Zahl eingibt.

## 1.4 Kapitel 4

### Übung 1

Beispielkonfiguration in der `php.ini`:

```
error_log = /c/Users/Philipp/php_errors.log;
error_reporting = E_ALL
display_errors = Off
```

Die Direktive `error_log` bestimmt den Pfad zu einer Log-Datei.

Über welche Fehlerstufen PHP berichtet, legt die Direktive `error_reporting` fest. Die Zuweisung der Konstante `E_ALL` steht dabei für »alle Fehlerarten« und unterstützt eine saubere Programmierung am besten, da alle Arten von Fehlern aufgedeckt und beseitigt werden können.

Die Direktive `display_errors` schaltet die direkte Fehleranzeige bei der Skriptausführung an oder aus (On oder Off). Beachten Sie, dass dies nur die direkte Ausgabe von Fehlern betrifft, nicht jedoch das Logging in eine Datei. Mit dem Ausschalten der Fehleranzeige werden die Fehler daher weiterhin geloggt.

Die Verwendung einer undefinierten Variable oder eine Division durch null verursacht Fehler verschiedener Fehlerstufen:

```
echo $undefinedVariable; // Warnung
echo 1/0; // Fataler Fehler (Division durch Null)
```

Überprüfen Sie nach Ausführung des Skripts die Log-Datei.

## Übung 2

Die maximal erlaubte Nutzung von Arbeitsspeicher für ein PHP-Skript bestimmt die `php.ini`-Direktive `memory_limit`. Mit der Funktion `ini_set()` kann der Vorgabewert aus der `php.ini`-Datei zur Laufzeit eines Skript verändert werden. Der veränderte Wert gilt aber nur bis zum Ende des Skripts, die `php.ini`-Datei wird dabei nicht verändert. Mit der Funktion `ini_get()` kann der aktuelle Wert einer Direktive zur Überprüfung ausgelesen werden.

## 1.5 Kapitel 5

### Übung 1

Der Funktionsaufruf `date('Y')` bestimmt zur Laufzeit dynamisch das aktuelle Jahr und wird zum Aufbau einer Zählschleife (`for`) vom Jahr 1900 bis zum aktuellen Jahr genutzt. Die Jahreszahl jeder Schleifen-Iteration dient als Argument für die Funktion `isLeapYear()`, welche den Algo-

rithmus zur Überprüfung auf ein Schaltjahr enthält. Die Funktion antwortet mit `true` oder `false`, je nachdem, ob es sich um ein Schaltjahr handelt oder nicht.

### Übung 2

Es gibt kein Formatzeichen für die `date()`-Funktion, um einen Wochentag bzw. Monatsnamen auf Deutsch zu erhalten. Die Lösung nutzt dazu die Formatzeichen »w« bzw. »n« aus, die den Wochentag (0-6) bzw. Monat (1-12) als Ganzzahlen darstellt. Diese Zahlen können als Array-Schlüssel eingesetzt werden, um die jeweilige Zahl auf den passenden deutschen Begriff aus den Hilfs-Arrays mit den deutschen Bezeichnungen abzubilden (bzw. im Programmier-Jargon: »zu mappen«).

### Übung 3

Mit einer `foreach`-Schleife lässt sich über die lange Liste an Zeitzonen iterieren, die die Funktion `timezone_identifiers_list()` liefert. In jeder Iteration wird zunächst die für das Skript aktuell gültige Zeitzone mit Hilfe der Funktion `date_default_timezone_set()` auf die jeweils nächste Zeitzone der Liste verstellt. Die darauffolgenden Aufrufe der Funktion `date()` beziehen sich dann auf die zuvor eingestellte Zeitzone.

Das Verstellen der Zeitzone zur Laufzeit hat keine Auswirkung auf den `php.ini`-Vorgabewert aus der Direktive `date.timezone`. Der Vorgabewert gilt zu Beginn jedes Skripts.

## 1.6 Kapitel 6

### Übung 1

Eine detaillierte Beschreibung des superglobalen Arrays `$_SERVER` finden Sie in der PHP-Dokumentation unter [www.php.net/manual/de/reserved.variables.server](http://www.php.net/manual/de/reserved.variables.server).



Beim Aufruf eines PHP-Skripts im Browser (d.h. über einen Webserver und nicht die Kommandozeile) enthält das `$_SERVER`-Array die vom Browser gesendeten HTTP-Header. Die Array-Schlüssel für HTTP-Header beginnen alle mit dem Präfix `HTTP_`, gefolgt vom Header-Namen; zum Beispiel `HTTP_USER_AGENT` für den Header `User-Agent` (dessen Wert den Browser bzw. das Betriebssystem des Benutzers angibt).

Beachten Sie, dass die Schlüssel der HTTP-Header im `$_SERVER`-Array einheitlich in Großbuchstaben und mit Unterstrichen statt Bindestrichen notiert werden.

### Übung 2

Die Interpretation einer HTTP-Antwort durch den Browser hängt von den Metadaten aus dem mitgelieferten HTTP-Header `Content-Type` ab. Sofern ein Skript diesen Header nicht selbst setzt, sendet PHP den Wert aus der `php.ini`-Direktive `default_mimetype`. Deren Standardwert lautet `text/html`. Der Browser wird also aufgefordert, den empfangenen Inhalt als HTML zu interpretieren. Falls ein Skript statt HTML binäre Daten eines Bilds sendet, diese per Header jedoch als HTML deklariert werden, zeigt der Browser nur Buchstabensalat.

Für die korrekte Anzeige eines Bilds muss dem Browser der passende Tipp gegeben werden, um welche Art von Daten (»Inhaltstyp«) es sich bei der Antwort handelt. Dies gelingt in der Lösung durch Setzen des HTTP-Headers `Content-Type` auf den MIME-Type `image/png` mit Hilfe der Funktion `header()`. Dies überschreibt den Standardwert aus der `php.ini` für die Laufzeit des Skripts. Der Browser wird die Daten dann als PNG-Bild interpretieren und korrekt darstellen.

## 1.7 Kapitel 7

### Übung 1

Das Formular im Lösungs-Quelltext gibt keine explizite Versandmethode mit dem `method`-Attribut an, daher wird es standardmäßig per GET-

Methode verschickt; die Datenübertragung erfolgt im Query-String. Das leere `action`-Attribut bedeutet, dass das Formular an das gleiche Skript abgeschickt wird. Oberhalb des Formulars lässt sich somit prüfen, ob die Daten im Query-String, d.h. im `$_GET`-Array, vorhanden sind. Sind sie das, wurde das Formular abgeschickt und das Skript kann die gewünschte Auswertung durchführen. Vergessen Sie bei der Ausgabe von Benutzereingaben nie die Ausgabemaskierung mit der Funktion `htmlspecialchars()`, um Cross-Site-Scripting (XSS) vorzubeugen.

Das HTML-Attribut `placeholder` kann einen Hinweis zu einer gewünschten Eingabe für ein Textfeld in einem Formular geben. Der Browser zeigt diesen Text im Feld leicht ausgegraut an, solange noch keine echte Eingabe in das Feld erfolgt ist. Beachten Sie, dass der `placeholder`-Wert kein Vorgabewert für das Feld ist, einen solchen müssten Sie mit dem `value`-Attribut angeben.

### Übung 2

Statt die Eingaben wie in der vorigen Übung im HTML auszugeben, werden sie in der Lösung nach Abschicken des Formulars als Text auf eine PNG-Bilddatei geschrieben. Nach dem Senden der Bilddaten mit der Funktion `imagepng()` an den Browser wird das Skript mit `exit()` beendet.

### Übung 3

Ein zusätzliches Formularfeld vom Typ `select` ermöglicht die Auswahl eines Werts aus einer Liste. Dabei steht jedes `option`-Element für einen möglichen Wert in der Liste. Beim Absenden wird nicht der im Browser sichtbare Wert übertragen, sondern der Wert aus dem `value`-Attribut des `option`-Elements; im Lösungs Quelltext `square` oder `circle`. Die Auswertung des Formulars überprüft, ob der übertragene Wert einem der beiden gültigen Werte entspricht. Mit einem gültigen Wert kann dynamisch der Dateipfad zur passenden Bilddatei (Quadrat oder Kreis) gebildet werden.

## Übung 4

Um das Formular per POST-Methode zu versenden, muss das `method`-Attribut des `form`-Elements auf den Wert `POST` gesetzt sein. Groß- bzw. Kleinschreibung spielt dabei keine Rolle. Bei der Verarbeitung des Formulars befinden sich die Formulardaten dann im superglobalen `$_POST`-Array.

## 1.8 Kapitel 8

### Übung 1

Die `include`-Anweisung bindet wie `require` eine Datei ein. Der Unterschied liegt im Fehlerverhalten, sollte die einzubindende Datei nicht existieren. Während `require` in diesem Fall das Skript mit einem fatalen Fehler beendet, kommt es bei `include` lediglich zu einer Warnung und das Skript läuft weiter. Da ein Skript kaum sinnvoll weiterarbeiten kann, wenn es eine Datei einbinden möchte, die nicht existiert, ist `require` grundsätzlich die bessere Wahl, um den Fehler zu verdeutlichen.

Ob eine Datei existiert, kann bei Bedarf vorab mit der Funktion `file_exists()` geprüft und behandelt werden:

```
$filepath = 'Pfad/zur/Datei';
if (file_exists($filepath)) {
    require $filepath;
} else {
    echo "Die benötigte Datei $filepath existiert nicht.";
}
```

### Übung 2

Eine CSV-Datei ähnelt einer Excel-Tabelle: Jede Zeile entspricht einem Datensatz, dessen einzelne Spalten bzw. Felder per Komma getrennt sind.

Die Funktion `file()` zerlegt den Inhalt der CSV-Datei zunächst an den Zeilenumbrüchen in ein eindimensionales Array; jedes Array-Element enthält dann eine Zeile bzw. einen Datensatz. Bei der Iteration über dieses Array wird jeder der Datensätze mit der Funktion `str_getcsv()` wiederum selbst an den Kommas in ein weiteres eindimensionales Array zerlegt; jedes Array-Element entspricht einer Spalte des Datensatzes. Enthält der Spalteninhalt selbst ein Komma, muss er in doppelte Anführungszeichen eingeschlossen werden.

Mit den weiteren Parametern der Funktion `str_getcsv()` kann die Interpretation der CSV-Struktur konfiguriert werden. So verwendet z.B. Microsoft Excel beim CSV-Export ein Semikolon statt Komma zur Trennung von Spalten. Der zweite Parameter ermöglicht die Umstellung des Trennzeichens:

```
$data = str_getcsv($line, ';')
```

### Übung 3

Die API-Dokumentation von OpenWeather unter

<https://openweathermap.org/current>

beschreibt ausführlich alle Möglichkeiten und Parameter zum Abruf von Wetterdaten. Die Beschreibungen der JSON-Antworten erklärt die Struktur der zurückgelieferten Daten.

Wichtig beim Zusammenstellen der Abfrage-URLs: Der Query-String einer URL unterliegt gewissen Einschränkungen; nicht alle Zeichen sind dort erlaubt. Betten Sie z.B. den Wert einer Benutzereingabe bzw. einer Variablen direkt als URL-Parameter ein, könnte ein ungültiger Query-String entstehen:

```
$url = 'https://...?q='.$_GET['city'];
```

Mit der Funktion `urlencode()` stellt PHP eine Funktion bereit, die einen String zur Verwendung als URL-Parameter passend kodiert. Unerlaubte Sonderzeichen werden dabei umgewandelt:

```
$url = 'https://...?q='.urlencode($_GET['city']);
```

Im folgenden Beispiel werden etwa das Leerzeichen und der deutsche Umlaut passend kodiert:

```
echo urlencode('Bad Wörrishofen');  
// => Bad+W%C3%B6rrishofen
```

### Übung 4

Ein HTML-Formular kann beliebig viele Dateiapload-Felder enthalten (input-Elemente vom Typ `file`). Damit die Dateien übertragen werden, muss das `form`-Element das Attribut `enctype` auf `multipart/form-data` setzen!

Beim Empfang der Formulardaten erzeugt jedes Dateiapload-Feld ein Array-Element im superglobalen `$_FILES`-Array mit dem Feldnamen als Schlüssel. Jedes dieser Elemente ist wiederum selbst ein immer gleich strukturiertes Array mit Metadaten zur hochgeladenen Datei.

Unter dem Schlüssel `error` findet sich in den Metadaten ein numerischen Fehlercode, der Auskunft gibt, ob der Upload fehlerfrei war (0) oder nicht (alle Codes größer 0). Für jeden Fehlercode gibt es vordefinierte Konstanten, die einen Vergleich erleichtern. Die PHP-Dokumentation beschreibt die Konstanten unter [www.php.net/features.file-upload.errors](https://www.php.net/features.file-upload.errors). Im Lösungs Quelltext sehen Sie z.B. die Verwendung der Konstante `UPLOAD_ERR_NO_FILE` (= Fehlercode 4), um zu erkennen, ob für ein Feld überhaupt eine Datei ausgewählt und übertragen wurde. Der Abgleich mit `UPLOAD_ERR_OK` (= Fehlercode 0) stellt sicher, dass ein Upload keine anderen Fehler verursacht hat und weiterverarbeitet werden kann.

Unter dem Schlüssel `tmp_name` findet sich in den Metadaten der temporäre Pfad zum Ablageort der hochgeladenen Datei. Er kann genutzt werden, um mit der Funktion `mime_content_type()` den MIME-Type der Datei herauszufinden.

Gültige Dateien verschiebt das Lösungsskript zur dauerhaften Speicherung an einen Ort außerhalb der Document Root, um sie vor öffentlichem Zugriff zu schützen. Zur Vermeidung von Namenskollisionen mit anderen hochgeladenen Dateien wird mit Hilfe von `uniqid()` ein zufälliger, eindeutiger Dateiname gebildet.

## 1.9 Kapitel 9

### Übung 1

Die Lösung erlaubt dem Benutzer die Zustimmung oder Ablehnung von Werbe-Cookies mit Hilfe von zwei *Radio-Buttons*. Radio-Buttons bilden eine Gruppe von Schaltknöpfen, von denen genau einer ausgewählt sein kann. Sie werden in HTML mit `input`-Elementen vom Typ `radio` ausgedrückt. Alle `input`-Elemente mit demselben Namen gehören zu einer Gruppe. Beim Absenden des Formulars wird der Wert aus dem `value`-Attribut des aktivierten Schaltknopfs unter dem Namen der Gruppe übertragen. Im Lösungs-Quellcode hat die Gruppe den Namen `consent` (engl. für *Zustimmung*). Je nach Auswahl des Benutzers wird der Wert 1 (für *Ja*) oder 0 (für *Nein*) per POST-Methode übertragen. Das Formular schickt die Daten an dasselbe Skript ab (leeres `action`-Attribut), die Auswertung findet daher im Skript oberhalb des Formulars statt.

Empfängt das Skript einen gültigen Wert (0 oder 1) im Array-Element `$_POST['consent']`, hat der Benutzer seine Entscheidung getroffen. Für Folgeaufrufe wird sie zur clientseitigen Speicherung in einem Cookie an den Browser geschickt. Anschließend wird der Browser mit dem `Location`-Header angewiesen, das Skript erneut zu laden. Dabei wird erstmalig das neue Cookie mitgesendet.

Falls das Skript ein vorhandenes `consent`-Cookie feststellt, muss das Formular nicht mehr angezeigt werden. Der Wert des Cookies lässt sich in einer `if`-Bedingung oder der Bedingung eines ternären Operators nutzen, um Entscheidungen auf Basis des Zustimmungstatus zu treffen bzw. den Status sichtbar zu machen (`0 = false = nicht zugestimmt`, `1 = true = zugestimmt`).

### Übung 2

Jeder Aufruf des Lösungsskripts startet mit `session_start()` in jedem Fall zunächst eine serverseitige Session. Der Benutzer sieht ein POST-Formular, um eine gewünschte Anzahl Tickets in seinen Einkaufswagen zu legen. Daneben wird der aktuelle Status des Einkaufswagens angezeigt sowie eine Möglichkeit, den Einkaufswagen zu leeren.

Die Anzahl an Tickets im Einkaufswagen wird in der Session gespeichert, d.h. in einer Datei auf dem Server (identifiziert durch die Session ID, die der Benutzer als Cookie erhält). So bleibt der Status des Einkaufswagens über verschiedene Seitenaufrufe und für die Dauer der Session bzw. bis zum Schließen des Browsers erhalten.

Schickt der Benutzer das Formular ab (auch mehrfach), wird immer die gesendete Ticketanzahl zur vorhanden Ticketanzahl in der Session addiert. Ist noch keine Anzahl vorhanden, wird sie zunächst mit 0 initialisiert. Der Status des Einkaufswagens bleibt dank der serverseitig gespeicherten Daten über die Seitenzugriffe hinweg erhalten.

Beim Klicken der Schaltfläche zum Leeren des Einkaufswagens wird die Ticketanzahl aus der Session gelöscht. Das Formular hierzu enthält nur eine Schaltfläche mit dem Namen `reset`. Auch Schaltflächen werden als Formulardaten übertragen. Das Skript kann am Vorhandensein von `$_POST['reset']` erkennen, ob der Benutzer ein Zurücksetzen veranlasst hat.

Nach der Verarbeitung der POST-Formulare wird der Browser per `Location`-Header angewiesen, die Seite neu zu laden. Dadurch wird ein er-

neutes Absenden des Formulars unterbunden, sollte der Benutzer die Seite anschließend im Browser manuell aktualisieren. Der aktuelle Skriptname für den Location-Header kann dynamisch aus `$_SERVER['PHP_SELF']` gewonnen werden. Auf diese Weise entsteht keine Abhängigkeit zum Dateinamen des Skripts (d.h. eine Umbenennung des Skripts zieht keine Änderung des Programmcodes nach sich).

## 1.10 Kapitel 10

### Übung 1

Für die Eigenschaften `name` und `description` ist der Datentyp `string` passend, für den Preis `float`.

Auf öffentliche Eigenschaften kann nach der Instanziierung eines Objekts von außen mit Hilfe des Objekt-Operators `->` schreibend und lesend zugegriffen werden.

Das Lösungsskript instanziert zwei Objekte der `Product`-Klasse, weist den Eigenschaften jeweils Beispieldaten zu und liest diese zur Ausgabe in die Tabelle weiter unten wieder aus.

### Übung 2

Auf private Eigenschaften kann nicht von außerhalb des Objekts zugegriffen werden. Innerhalb von Methoden ermöglicht jedoch die spezielle Variable `$this` den Zugriff. Die Konstruktor-Methode kann daher die gewünschten Werte für die Eigenschaften als Parameter entgegennehmen und mit Hilfe von `$this->Eigenschaft` setzen. Dabei kann zuvor auch beliebiger Code zur Überprüfung auf gültige Eingaben durchlaufen werden. Bei ungültigen Eingaben wirft die `Product`-Klasse im Lösungsskript eine Ausnahme und die Erstellung eines Objekts würde fehlschlagen.

Der Zugriff auf die privaten Eigenschaften lässt sich nun durch Methoden kontrollieren. Zum Lesen der Eigenschaften wird der direkte Zugriff durch den Aufruf von passenden öffentlichen Methoden ersetzt. Die Methoden



innerhalb der Klasse haben mit der speziellen Variable `$this` Zugriff auf die Eigenschaften. Dabei können die Methoden auch Logik einbauen, um etwa den Preis direkt als formatierten String auszugeben. Davon profitieren alle Nutzer eines Objekts, denn sie müssen diese Logik nicht außerhalb des Objekts bei jeder Nutzung wiederholen.

### Übung 3

Ein Objekt der Klasse `DateTimeZone` repräsentiert eine Zeitzone. Die gewünschte Zeitzone wird beim Erstellen des Objekts an den Konstruktor übergeben. Eine Namensliste der gültigen Zeitzonen liefert die Funktion `timezone_identifiers_list()` oder die PHP-Dokumentation unter [www.php.net/manual/de/timezones](http://www.php.net/manual/de/timezones).

Ein Zeitzonen-Objekt kann bei der Erstellung eines `DateTime`-Objekts eingesetzt werden, um die Zeitzone des repräsentierten Datums festzulegen. Die Zeitzone kann später auch durch Zuweisung eines anderen Zeitzonenobjekts geändert werden – wie beim Verstellen einer Uhr.

Die Instanziierung des `DateTime`-Objekts im Lösungsskript mit dem gewünschten Datum »erster Tag des nächsten Monats mittags« erfolgt mit der relativen Zeitangabe »first day of next month 12:00«. Eine umfassende Beschreibung der möglichen Schlüsselwörter für relative Datumsangaben beschreibt die PHP-Dokumentation unter [www.php.net/manual/de/datetime.formats.relative](http://www.php.net/manual/de/datetime.formats.relative).

## 1.11 Kapitel 11

### Übung 1

Das SQL-Lösungsskript `employees.sql` demonstriert alle nötigen Befehle in der passenden Reihenfolge. Zunächst wird sichergestellt, dass eine leere Datenbank `company` existiert und für die folgenden Befehle mit `USE` zur Verwendung ausgewählt ist.

Für die Speicherung des Geburtsdatums eignet sich eine Spalte `date_of_birth` vom Typ `DATE`; für die Postleitzahl genügt eine auf fünf Zeichen begrenzte Spalte `zip` vom Typ `CHAR`. Für das Geschlecht kann eine Spalte `gender` vom Typ `ENUM` mit den drei zulässigen Werten `m`, `f` oder `d` verwendet werden. Für eine Mitarbeiter-Datenbank könnten alle Daten zwingend erforderlich sein, sie sind daher in der Lösung »nicht nullable« (`NOT NULL`). Bei den folgenden `INSERT`-Befehlen zum Einfügen von Beispieldatensätzen müssen daher immer passende Werte für die drei neue Spalten angegeben werden.

Zur Abfrage *aller* Spalten wird im `SELECT`-Befehl das Asterisk-Zeichen (\*) statt einzelner Spaltenangaben verwendet. Ohne `WHERE`-Klausel erhalten Sie *alle* Datensätze der Tabelle.

### Übung 2

Die `SELECT`-Abfrage aus Übung 1 wird mit `ORDER BY`- und `WHERE`-Klauseln entsprechend den Anforderungen angepasst.

### Übung 3

Bei der Aktualisierung einer Spalte kann der zugewiesene Wert statisch sein:

```
UPDATE employee SET salary = 50000
```

Es sind auch Berechnungen möglich, hier die Zuweisung des Werts 50.000 plus 10%:

```
UPDATE employee SET salary = 50000 * 1.1
```

Für eine dynamische Zuweisung kann der aktuelle Spaltenwert eines Datensatzes (d.h. der Wert vor der Aktualisierung!) als Variable fungieren. Im Folgenden wird der aktuelle Wert der `salary`-Spalte verwendet, um der `salary`-Spalte einen neuen, um 10% erhöhten Wert zuzuweisen:

```
UPDATE employee SET salary = salary * 1.1
```

Die Abfrage führt die Erhöhung für jeden einzelnen Datensatz durch, da es keine einschränkende `WHERE`-Klausel gibt.

## 1.12 Kapitel 12

Die beiden Übungen dieses Kapitels erfordern zunächst das Aufsetzen der Datenbank `book` und die darin enthaltene Tabelle `movie`. Importieren Sie dazu die vorbereitete SQL-Datei `movie.sql` aus dem Code-Verzeichnis zum Kapitel über den Reiter `IMPORTIEREN` in `phpMyAdmin`.

### Übung 1

Folgender SQL-Befehl erfüllt die Anforderungen:

```
SELECT * FROM book.movie
WHERE genre in ('Action', 'Drama') AND rating > 7
ORDER BY `year` DESC;
```

Nach Ausführung des SQL-Befehls in `phpMyAdmin` über den Reiter `SQL` in der Tabelle `book.movie` folgen Sie dem Link `EXPORTIEREN` am Ende der Ergebnisseite. Wählen Sie bei den Export-Einstellungen das Format `PDF` und klicken Sie `OK`, um das PDF mit den Ergebnisdatensätzen herunterzuladen.

### Übung 2

Die Aggregatfunktion `AVG(Spalte)` bildet den Durchschnittswerts aus allen Werten der *Spalte*.

`ROUND(Wert, 1)` rundet *Wert* auf eine Stelle genau.

`CONCAT(String1, String2, ...)` verknüpft *String1*, *String2*, usw. zu einem einzigen Ergebnis-String.

`DATE_FORMAT(Datum, Format)` formatiert das *Datum* im *Format*. Im Gegensatz zur PHP-Funktion `date()` muss in SQL jedem Formatzei-

chen ein %-Zeichen vorangestellt werden, z.B. %H:%i für *Stunde:Minuten*.

Die Aggregatfunktion `COUNT()` zählt Datensätze einer `SELECT`-Abfrage. `COUNT(*)` zählt dabei unabhängig von einer Spalte alle Datensätze, auf die die Abfrage zutrifft, `COUNT(Spalte)` zählt nur die Datensätze, in denen *Spalte* zugleich nicht `NULL` ist.

Die `WHERE`-Klausel »`Spalte LIKE '%Suchwort%'`« findet alle Datensätzen, in denen das *Suchwort* in der *Spalte* an beliebiger Stelle im Inhalt auftritt. Beachten Sie, dass MySQL für `LIKE`-Suchen mit einem Prozentzeichen zu Beginn eine vollständige Durchsuchung der Tabelle durchführen muss, weshalb eine solche Suche mit zunehmender Tabellengröße langsam wird. Eine Lösung bietet die »Volltextsuche«, ein fortgeschrittenes MySQL-Feature.

### 1.13 Kapitel 13

Alle Übungen dieses Kapitels verwenden eine Datenbank mit dem Namen `book`. Das im Buchkapitel vorgestellte Skript `_mysql-connect.php` wird zur Herstellung der Datenbank-Verbindung eingesetzt.

#### Übung 1

Die Lösung setzt die Anlage der Tabelle `user` (und einiger Beispieldatensätze) in der Datenbank `book` voraus, siehe Abschnitt 13.3 im Buch.

Nach Herstellung der Datenbankverbindung führt das Lösungsskript den passenden `SELECT`-Befehl mit der `PDO::query()`-Methode aus. Die Methode `PDO::fetchAll()` liest die Ergebnisdatensätze als Array in die Variable `$users` ein. Jedes Arrayelement entspricht einem Ergebnisdatsatz.

Zur Anzeige der Benutzer iteriert das Skript über die Datensätze im `$users`-Array und erzeugt für jeden eine Tabellenzeile. Die Spaltenschlüssel entsprechen den Spaltennamen aus der Datenbanktabelle.

### Übung 2

Statt das Passwort bei der Registrierung eines neuen Benutzers im Skript `register.php` im Klartext in die Datenbank zu schreiben, verwendet die überarbeitete Version zuvor die Funktion `password_hash()`, um das Passwort nach dem aktuell empfohlenen Einweg-Algorithmus zu verschlüsseln (zu »hashen«). Nur der Hash landet in der Datenbank; von diesem kann kein Rückschluss auf das Klartext-Passwort gezogen werden.

Beim Login im Skript `login.php` ändert sich das Vorgehen zum Abgleich des Passworts. Wird genau ein Datensatz mit dem passenden Benutzernamen gefunden, kommt die Funktion `password_verify()` zum Einsatz, um das beim Login eingegebene Klartext-Passwort mit dem verschlüsselten Passwort des Datensatzes abzugleichen. PHP verschlüsselt dabei das eingegebene Klartext-Passwort auf die gleiche Weise wie zuvor bei der Registrierung und vergleicht die beiden verschlüsselten Werte. Bei einer Übereinstimmung ist der Login erfolgreich.

Das Klartext-Passwort muss zu keinem Zeitpunkt abgespeichert werden.

### Übung 3

Die Definition der Tabelle `user` im Skript `create-user-table.php` wird mit der `DATETIME`-Spalte `last_login` erweitert. Die Spalte erlaubt den Wert `NULL`, da vor dem ersten Login noch kein Datum bekannt sein kann.

Löschen Sie gegebenenfalls eine vorhandene Tabelle `user` und führen Sie das Skript `create-user-table.php` erneut aus – oder ergänzen Sie die Spalte manuell über phpMyAdmin, um vorhandene Beispieldaten nicht zu verlieren.

Zur Speicherung des aktuell letzten Login-Datums wird nach jedem erfolgreichen Login in der Datei `login.php` ein SQL `UPDATE`-Befehl ausgeführt, der mit Hilfe der SQL-Funktion `NOW()` das aktuelle Datum in die neue Spalte `last_login` des Benutzerdatensatzes in der `user`-Tabelle schreibt.

Bei der Auflistung aller Benutzer im Skript `list-users.php` wird die neue Spalte berücksichtigt. Die SQL-Funktion `DATE_FORMAT()` fragt das letzte Login-Datum dabei gleich im passenden Format für die Ausgabe ab.

## 1.14 Kapitel 14

### Übung 1

Wie für jeden Anwendungsfall bindet auch das Skript `article-edit.php` zunächst die Datei `_initialize.php` ein, um alle Grundfunktionen der Blog-Anwendung zur Verfügung zu haben.

Falls die anschließende Überprüfung des Login-Status negativ ausfällt, wird der Benutzer mit einer entsprechenden Meldung auf die Startseite umgeleitet. Nur eingeloggte Benutzer dürfen Artikel bearbeiten.

Wie in den anderen Skripten folgt dann die individuelle Logik für den Anwendungsfall. Das Array `$formErrors` wird leer initialisiert, um mögliche Validierungsfehler des Formulars aufzunehmen.

Beim Bearbeiten eines Artikels sind zwei Fälle zu unterscheiden:

#### Artikel wurde initial zur Bearbeitung aufgerufen

Ein Artikel wurde über die Detailansicht (`article.php`) zur Bearbeitung ausgewählt; er wird durch die in der URL übergebene Artikel-ID identifiziert. Das Skript entscheidet sich für diesen Fall, wenn die aufrufende Methode *nicht* POST lautet; im Skript bedeutet dies das Durchlaufen des `else`-Zweiges. Anhand der Artikel-ID können die aktuellen Artikel-Daten aus der Datenbank abgerufen und jede Spalte in entsprechende Variablen geladen werden.

Sind die Daten geladen, beginnt die Ausgabe des HTML-Codes für die Webseite mit dem Bearbeitungsformular. Nach der Datei `_header.php` zur Anzeige des gewohnten Kopfbereichs folgt die Einbindung von `_article-form.php`. Diese Datei ist bereits von der Artikel-Erstellung bekannt und kann hier geschickt wiederverwendet werden. Dank der zu-

vor gesetzten Variablen mit den Artikeldaten sind alle Formular-Felder mit den aktuellen Inhalten vorbelegt. Abschließend wird der gewohnte Fußbereich der Website mit `_footer.php` eingebunden.

Der Benutzer kann mit der Bearbeitung des Artikels im Browser beginnen und das Formular am Ende per POST-Methode an dasselbe Skript zur Auswertung abschicken.

### Das Formular zur Artikel-Bearbeitung wurde abgeschickt

Dieser Fall tritt ein, wenn das Formular mit bearbeiteten Artikeldaten abgeschickt wurde. Die Übertragungsmethode lautet dabei POST und das Skript entscheidet sich für den `if`-Zweig seiner Logik. Im Vergleich zur Anlage eines neuen Artikels übermittelt das Formular in einem versteckten Feld (`input`-Element vom Typ `hidden`) die Artikel-ID des existierenden Artikels. Sind die übermittelten Daten aus dem `$_POST`-Array gültig, kann der passende Artikel mit Hilfe der Artikel-ID in der Datenbank aktualisiert und der Benutzer zur Artikelansicht weitergeleitet werden. Gab es Fehler bei der Validierung der übermittelten Daten, wird das Formular mit den im Array `$formErrors` gesammelten Fehlermeldungen erneut angezeigt; der Benutzer kann dann nachbessern und das Formular erneut absenden.

Zur Validierung der Formulardaten wird die von der Artikelerstellung bekannte Funktion `validateArticle()` wiederverwendet.

## Übung 2

Die Lösung dieser Übung benötigt eine Installation des Composer-Pakets `dompdf/dompdf`. Installieren Sie das Paket auf der Kommandozeile im Verzeichnis mit dem Lösungsskript:

```
> composer require dompdf/dompdf
```

Bei der Installation werden die Dateien `composer.json` und `composer.lock` sowie das Verzeichnis `vendor/` erzeugt. Im Verzeichnis `vendor/` befinden sich die PHP-Dateien aller installierten Pakete.

Ein Paket kann wiederum selbst andere Pakete erfordern, die ebenfalls automatisch mitinstalliert werden.

Die Erzeugung des PDFs folgt denselben Schritten wie im Skript `article-pdf.php` des Blog-Projekts. Zur Umschaltung auf das Querformat (»landscape«) wird eine Zeile zur Einstellung des Papierformats ergänzt:

```
$pdf->setPaper('A4', 'landscape');
```

Die Dokumentation von `dompdf/dompdf` finden Sie unter <https://github.com/dompdf/dompdf/wiki>.

### Übung 3

Die Lösung dieser Übung benötigt eine Installation des Composer-Pakets `symfony/mailer`. Installieren Sie das Paket auf der Kommandozeile im Verzeichnis mit dem Lösungsskript:

```
> composer require symfony/mailer
```

Bei der Installation werden die Dateien `composer.json` und `composer.lock` sowie das Verzeichnis `vendor/` erzeugt. Im Verzeichnis `vendor/` befinden sich die PHP-Dateien aller installierten Pakete. Ein Paket kann wiederum selbst andere Pakete erfordern die ebenfalls automatisch mitinstalliert werden.

Der Versand der E-Mail folgt denselben Schritten wie im Skript `article-feedback.php` des Blog-Projekts. Zur Ergänzung von CC-Empfänger und HTML-Inhalt werden passende Methoden des `Email`-Objekts aufgerufen:

```
$email = (new Symfony\Component\Mime\Email())  
    // ...  
    ->cc('m.name@me.com')  
    ->html('<h1>Wunderbar!<h1>');
```



Die Dokumentation des Pakets `symfony/mailer` finden Sie unter <https://symfony.com/doc/current/mailer.html>.