

Carsten Möhrke

Inkl.  
PHP 5.4

# Besser PHP programmieren

Handbuch professioneller PHP-Techniken

Design Patterns

PHPUnit

Ajax

Subversion

Debugging

Sicherheit

Errorhandling

Zend Studio

jQuery

CouchDB

Zend Framework

Performance-Optimierung

MVC-Architektur

u. v. m.



4., aktualisierte und  
erweiterte Auflage

Galileo Computing 

# Inhalt

Vorwort zur vierten Auflage .....	13
-----------------------------------	----

## 1 Die Arbeit mit PHP ..... 15

1.1	Lernen Sie Ihr Arbeitsgerät kennen .....	15
1.2	Der Editor .....	17
1.3	Eclipse mit PDT .....	19
1.3.1	Der Editor .....	22
1.3.2	Konfiguration .....	29
1.4	Zend Studio .....	31
1.5	Der Server .....	32
1.5.1	Serverkonfiguration .....	36
1.5.2	Nutzung des Servers .....	38
1.6	Subversion .....	40
1.6.1	Der Server .....	41
1.6.2	Der Client .....	43
1.7	Der PEAR-Installer .....	51

## 2 Datentypen und -konvertierung ..... 53

2.1	Datentypen in PHP .....	53
2.1.1	Boolean .....	53
2.1.2	Integer .....	54
2.1.3	Float .....	56
2.1.4	String .....	60
2.2	Typkonvertierung .....	64
2.3	Arrays .....	68
2.3.1	Allgemeines zu Arrays .....	68
2.3.2	Vergleich von Arrays .....	68
2.3.3	Ausgabe von Arrays .....	69
2.3.4	Kombinieren von Arrays .....	73
2.3.5	Verarbeiten von Arrays .....	75
2.3.6	Übereinstimmungen und Unterschiede in Arrays .....	82
2.4	Objekte .....	84
2.5	Spezielle Datentypen .....	85
2.5.1	NULL .....	85
2.5.2	Resource .....	86

<b>3 Programmierstil .....</b>	<b>87</b>
3.1 HTML in PHP oder PHP in HTML? .....	88
3.1.1 Template-Systeme .....	91
3.2 Allgemeines zur Programmierung .....	92
3.2.1 Disziplin .....	93
3.2.2 Verständlichkeit .....	93
3.2.3 Alternative PHP-Syntax .....	95
3.2.4 Kommentare .....	96
3.2.5 Vermeiden Sie »Magic Numbers« .....	99
3.2.6 ToDos, Fallstricke und andere Probleme .....	100
3.2.7 Halten Sie Ordnung .....	101
3.3 Quelltextformatierung .....	102
3.3.1 Globale Struktur .....	102
3.3.2 Klammerung .....	103
3.3.3 Einrückungen .....	105
3.3.4 Verschachtelte Funktionsaufrufe .....	105
3.3.5 SQL und JavaScript .....	106
3.3.6 Komplexe Bedingungen .....	107
3.4 Namensgebung .....	108
3.4.1 Abkürzungen .....	109
3.4.2 Namen für Variablen und Konstanten .....	110
3.4.3 Namen für Funktionen und Klassen .....	113
3.5 Kontrollstrukturen .....	115
3.5.1 Bedingungen .....	115
3.5.2 Fallunterscheidungen .....	120
3.5.3 Der ternäre Operator .....	123
3.5.4 Die Arbeit mit Schleifen .....	125
3.5.5 Die Sache mit dem goto .....	128
3.6 Is it a bug or is it a feature? .....	129
3.6.1 Die Inkrement-Operatoren .....	130
3.6.2 Die Funktion empty() .....	131
3.6.3 Groß-/Kleinschreibung .....	132
3.6.4 Parameter von Funktionen .....	133
3.6.5 Neue PHP-Versionen .....	134
3.7 Refactoring – Kampf dem historisch gewachsenen Code .....	135
3.7.1 Funktionen/Methoden umbenennen (Rename Method) .....	135
3.7.2 Code in Funktion auslagern (Extract Method) .....	136
3.7.3 Bedingungen auslagern (Decompose Conditional) .....	136

3.7.4	Wiederholte Befehle aus Bedingungen zusammenfassen (Consolidate Duplicate Conditional Fragments) .....	138
3.8	Der PEAR-Coding-Standard (PCS) .....	139
3.8.1	Einrücken und Zeilenlänge .....	139
3.8.2	Kontrollstrukturen .....	139
3.8.3	Funktionsaufrufe .....	140
3.8.4	Funktionsdeklaration .....	140
3.8.5	Kommentare und Dokumentation .....	140
3.8.6	Code einbinden .....	141
3.8.7	PHP-Tags .....	141
3.8.8	Namenskonventionen .....	141
3.8.9	Dateiformat .....	142
<b>4</b>	<b>Modularisierung von Code .....</b>	<b>143</b>
4.1	Arbeiten mit externen Bibliotheken .....	143
4.2	Funktionen .....	146
4.2.1	Design von Funktionen und Methoden .....	149
4.3	Objektorientierung .....	157
4.3.1	Zugriff auf Objekte .....	158
4.3.2	Deklaration von Klassen .....	159
4.3.3	Fortgeschrittene objektorientierte Programmierung .....	180
4.3.4	Entwurfsmuster .....	221
<b>5</b>	<b>Error Handling .....</b>	<b>245</b>
5.1	Der @-Operator .....	251
5.2	Eigene Error Handler .....	252
5.2.1	Anderer Leute Fehler .....	258
5.2.2	Fehlermanagement .....	259
5.3	Error Handling in Bibliotheken .....	268
5.3.1	Error Handling bei Funktionsbibliotheken .....	268
5.3.2	Error Handling in Klassenbibliotheken .....	273
5.4	Exception Handling .....	278
5.5	Fehlerdokumente .....	283
<b>6</b>	<b>Professionelle Bibliotheken .....</b>	<b>291</b>
6.1	Allgemeines zu Frameworks und Klassenbibliotheken .....	292
6.1.1	Auswahl eines Frameworks oder einer Bibliothek .....	292
6.2	Smarty .....	300

6.2.1	Modifikatoren .....	308
6.2.2	Funktionen .....	316
6.2.3	Caching .....	323
6.3	Frameworks .....	325
6.3.1	Zend Framework .....	325
6.3.2	Der Front Controller .....	333
6.3.3	Der Action Controller .....	334
6.3.4	Nutzung von Views .....	337
6.3.5	Die Mitspieler im Einzelnen .....	340
6.3.6	Übergabe von Werten .....	341
6.3.7	Error Handling .....	344
6.3.8	Hilfreiche Techniken .....	348
6.3.9	Das Model .....	350

## 7 Qualitätssicherung ..... 387

7.1	Im Vorfeld .....	387
7.2	Konzeption einer Anwendung .....	388
7.2.1	Lufthansa Flug 2904 .....	391
7.2.2	Die manövrierunfähige USS Yorktown .....	392
7.2.3	Mock-ups .....	393
7.3	Das Pflichtenheft .....	397
7.4	Qualitätsmerkmale .....	399
7.5	Reviews .....	400
7.6	Automatisierte Qualitätssicherung .....	401
7.6.1	PHP Codesniffer .....	402
7.6.2	PHP Copy and Paste Detector .....	405
7.6.3	PHP Mess Detector .....	406
7.6.4	Testgetriebene Entwicklung mit PHPUnit .....	412
7.6.5	Testen mit Selenium .....	423
7.6.6	Lasttests .....	429
7.7	Debugging .....	437
7.7.1	PHP-interne Debug-Features .....	437
7.7.2	Eigene Debug-Routinen .....	442
7.7.3	Professionelle Debugger .....	447
7.8	Testen .....	454
7.8.1	Personal .....	454
7.8.2	Vorgehensweise .....	456

<b>8</b>	<b>Dokumentation .....</b>	<b>459</b>
8.1	Anforderungen an eine Dokumentation .....	460
8.2	Programmablaufpläne und Struktogramme .....	463
8.3	phpDocumentor .....	468
8.3.1	Die wichtigsten Tags .....	476
8.3.2	Kommandozeilenoptionen .....	482
<b>9</b>	<b>Praxislösungen für den Programmieralltag .....</b>	<b>485</b>
9.1	Elementare Datenstrukturen und Algorithmen .....	485
9.1.1	Mengen .....	485
9.1.2	Queues .....	487
9.1.3	Stacks .....	488
9.1.4	Verkettete Listen .....	488
9.1.5	Bäume und Rekursionen .....	493
9.1.6	Nested Sets .....	506
9.2	Zeichensätze .....	512
9.2.1	Warum eigentlich Zeichensätze? .....	513
9.2.2	Die Geschichte der Zeichensätze .....	514
9.2.3	Zeichensätze bei Webseiten .....	519
9.2.4	PHP und Zeichensätze .....	522
9.2.5	MySQL und Zeichensätze .....	530
9.3	Interaktion mit Benutzern .....	533
9.3.1	Aufbau von Formularen .....	533
9.3.2	Wertübernahme aus Formularen .....	537
9.3.3	Mehrfaches Abschicken von Formularen .....	541
9.3.4	Prüfen von Benutzereingaben .....	544
9.3.5	Formulare mit Ajax .....	548
9.4	Reguläre Ausdrücke .....	570
9.4.1	Delimiter .....	571
9.4.2	Zeichenklassen .....	573
9.4.3	Quantifier .....	575
9.4.4	Anker .....	576
9.4.5	Submuster .....	577
9.4.6	Backreferences .....	578
9.4.7	Lookaround .....	578
9.4.8	Bedingte Ausdrücke .....	581
9.4.9	Gier .....	582
9.4.10	Optionen .....	582
9.4.11	PCRE-Funktionen .....	584

9.5	Arbeit mit Dateien .....	588
9.5.1	Unterschiede bei Dateien .....	590
9.5.2	CSV-Dateien .....	595
9.5.3	Locks auf Dateien .....	599
9.6	E-Mails .....	605
9.6.1	Allgemeines zu E-Mails .....	605
9.6.2	Empfänger der E-Mail .....	607
9.6.3	Header-Felder .....	608
9.6.4	MIME-E-Mails .....	613
9.6.5	Anhänge komprimieren .....	623
9.6.6	E-Mails verschlüsseln .....	625
9.7	Sicherheit .....	632
9.7.1	Bleiben Sie auf dem Laufenden .....	633
9.7.2	Fertige Lösungen .....	634
9.7.3	Sessions .....	636
9.7.4	Globals .....	645
9.7.5	Verschiedene Angriffsarten .....	651
9.7.6	Passwörter .....	675
9.7.7	CAPTCHAs .....	684
9.7.8	Altersüberprüfung .....	696
9.7.9	Schutz von E-Mail-Adressen .....	702
9.8	Shared Memory .....	706
9.9	Installationsprogramme .....	718
9.9.1	Installationsvoraussetzungen .....	718
9.9.2	Übernahme von Werten .....	725
9.9.3	Konfigurationsdateien .....	727
9.9.4	Installation von Komponenten .....	730
9.9.5	Tabellen .....	734
9.9.6	Serverzeit .....	736
9.9.7	Grafiken .....	743
9.9.8	Uninstall .....	746
9.10	Internationalisierung/Lokalisierung .....	749
9.10.1	Mehrsprachige Texte .....	749
9.10.2	Datums- und Zahlenformate .....	752
9.11	Performance-Tuning .....	757
9.11.1	Performance-Bremsen finden .....	762
9.11.2	String-Verarbeitung .....	773
9.11.3	Statische Seiten generieren .....	775
9.11.4	Datenbankabfragen .....	779
9.11.5	Cache-Systeme .....	784

9.12	Genau rechnen .....	792
9.12.1	BCMath .....	793
9.13	Arbeit mit Datenbanken .....	795
9.13.1	Allgemeines .....	795
9.13.2	Transaktionsorientierung .....	805
9.13.3	phpMyAdmin .....	807
9.13.4	ODBC .....	813
9.13.5	Volltextsuche in einer MySQL-Datenbank .....	818
9.13.6	Datenbank-Performance .....	820
9.13.7	Stored Procedures, Stored Functions und Trigger .....	832
9.13.8	NoSQL-Datenbanken .....	847
	Inhalt der DVD .....	865
	Index .....	867



*If it doesn't fit, use a bigger hammer.*  
– Miles O'Brian, *Star Trek*

## 2 Datentypen und -konvertierung

### 2.1 Datentypen in PHP

Im Gegensatz zu vielen anderen Programmiersprachen unterstützt PHP leider keine Variablendeklaration, und der Programmierer hat auch nicht die Möglichkeit, einer Variablen einen Typ zuzuweisen. PHP ermittelt selbst, welchen Typ eine Variable haben muss, und führt ein automatisches »Type Casting« durch. Das heißt, eine Variable kann automatisch von einem Typ in einen anderen konvertiert werden. Das führt dazu, dass sich viele Entwickler keine Gedanken über den Datentyp machen. Frei nach dem Motto »PHP wird's schon richten« gehen sie davon aus, dass die Konvertierung automatisch erfolgt und keine unangenehmen Nebeneffekte haben wird. Leider ist das nicht immer so.

Aber ich möchte die implizite Konvertierung nicht nur verteufeln. Sie macht vieles einfacher. Und wenn Sie mal mit einer streng typisierten Sprache gearbeitet haben, dann wissen Sie, wie nervig es sein kann, wenn man Datentypen ständig konvertieren muss. Bitte vergessen Sie aber nie, dass die implizite Konvertierung schnell zu einem Problem werden kann.

PHP kennt vier skalare Datentypen:

- ▶ Boolean
- ▶ Integer
- ▶ Float (Double)
- ▶ String

#### 2.1.1 Boolean

Bei Boolean handelt es sich um einen Datentyp, der nur zwei Werte unterstützt, und zwar `true` (wahr) und `false` (falsch). Häufig wird `true` mit dem Wert 1 und `false` mit dem Wert 0 gleichgesetzt, was so aber nicht korrekt ist. Wird ein Boolescher Wert in einer Berechnung genutzt, konvertiert PHP ein `true` allerdings in 1

und `false` in `0`. Das funktioniert natürlich auch in die andere Richtung. Die Zahl `0` kann in ein `false` konvertiert werden, wohingegen alle anderen Werte als `true` interpretiert werden. Aber Achtung: eine Besonderheit bei Booleschen Werten führt schnell zu Irritationen. Stellen Sie sich vor, Sie wollen zum Debuggen »mal gerade schnell« den Wert einer Variablen auslesen. Nichts einfacher als das:

```
$a = true;
$b = false;
echo '$a: ' . $a;
echo "<br>";
echo '$b: ' . $b;
/* Ausgabe im Browser:
$a: 1
$b:
*/
```

Lassen Sie Boolesche Werte direkt per `echo` ausgeben, oder konvertieren Sie sie in einen String, dann wird also nur das `true` in eine `1` konvertiert. Das `false` hingegen resultiert in einem Leer-String.

Boolesche Werte werden häufig als Rückgabewerte von Funktionen genutzt.

### 2.1.2 Integer

Bei dem Datentyp `Integer` handelt es sich um ganzzahlige Werte, also Zahlen ohne einen Nachkommaanteil. Wie groß ein Integer-Wert sein darf, ist heutzutage nicht mehr ganz so einfach zu beantworten. Das hängt nämlich von dem Betriebssystem ab, auf dem PHP ausgeführt wird. Nutzen Sie ein 32-Bit-System, dann kann der Wert zwischen `-2.147.483.648` und `2.147.483.647` liegen. Nutzen Sie allerdings ein 64-Bit-System, dann steht Ihnen der Wertebereich von `-9.223.372.036.854.775.808` bis `9.223.372.036.854.775.807` zur Verfügung. Wie groß der Wertebereich ist, den Ihr System abdeckt, können Sie der Konstante `PHP_INT_MAX` entnehmen. Sie enthält den maximalen Wert. Der minimale Wert entspricht dem maximalen mit negativem Vorzeichen minus 1.

PHP nutzt hierbei einen sogenannten »signed Integer«. Das heißt, dass sowohl der negative als auch der positive Zahlenbereich abgebildet werden kann. Ein »unsigned Integer«, also eine Integer-Darstellung, die nur positive Werte unterstützt, kann in PHP nicht genutzt werden.

Bei der Wertzuweisung sollten Sie beachten, dass PHP auch das hexadezimale und das oktale Zahlensystem unterstützt. Eine Zahl, die mit einer `0` beginnt, wird als oktal gewertet und ein Wert, der mit einem `0x` beginnt, als hexadezimale Zahl. Dies kann schnell zu einem unerwarteten Verhalten führen:

```
$oktal = 0815; // beliebte Zahl zum Testen
echo $oktal; // gibt 0 aus, da 8 keine gültige oktale Zahl ist
```

```
$oktal2 = 042; // auch eine beliebte Zahl
echo $oktal2; // gibt 34 aus
```

```
$hex = 0x11;
echo $hex; // gibt 17 aus
```

Achten Sie also auch bei Werten, die Sie »nur mal gerade zum Testen« eingeben, sehr genau darauf, dass Sie nicht mit einer führenden Null arbeiten.

Ab PHP 5.4 haben Sie übrigens auch die Möglichkeit, Zahlen binär anzugeben. Das heißt, Sie können Integer-Werte mit Nullen und Einsen angeben, wenn Sie der Zahl 0b voranstellen.

```
<?php
$int = 0b10;
echo $int; // gibt 2 aus
?>
```

Hier wird 2 ausgegeben, weil das Bit an der Stelle 0 nicht gesetzt und das Bit an der Stelle 1 gesetzt ist. Das heißt, die Zahl wird von rechts nach links interpretiert, und die 0 steht dafür, dass das Bit  $2^0$  nicht gesetzt ist. Die 1 bedeutet, dass das Bit an der Stelle  $2^1$  gesetzt ist. Hätte ich 0b110 angegeben, dann würde sich die Zahl 6 ( $2^2 + 2^1 + 0$ ) ergeben.

Jetzt fragen Sie sich sicher, wozu das gut ist. Wenn Sie gerade einfach mal nicht an Zahlen denken, sondern an Schalter, dann kann man mit einer 1 »eingeschaltet« (oder zulässig) symbolisieren und mit einer 0 »ausgeschaltet« (oder unzulässig) darstellen. Wollten Sie beispielsweise ein kleines Rechtesystem aufbauen, dann könnten Sie mit dem Bit an der Stelle 1 symbolisieren, dass jemand das Recht hat, einen Artikel zu schreiben, wohingegen das Bit an der Stelle 0 dafür stünde, dass jemand lesen darf. Das heißt, wenn jemand 0b11 hat, dann dürfte er lesen und schreiben. Das hört sich zunächst umständlich an. Wenn Sie nun herausfinden wollten, was jemand darf, dann müssten Sie immer mit Integer-Werten vergleichen. Wenn jemand ein Recht  $>1$  und  $<4$  hätte, dann dürfte er lesen.

Das geht aber auch einfacher, wenn Sie den Bit-Operator & nutzen. Verknüpfen Sie mit ihm zwei Bitfolgen, dann gibt er an den Stellen, an denen Sie bei Bitfolgen eine 1 hatten, eine 1 zurück. Die folgenden Beispiele verdeutlichen das System:

0010 & 0011 = 0010

1111 & 1000 = 1000

0000 & 1111 = 0000

Mit diesem System können Sie jetzt also erkennen, an welcher Stelle ein Bit gesetzt ist. Konvertieren Sie das Ergebnis einer solchen Operation jetzt in einen Booleschen Wert, dann ergibt sich für 0 ein `false` und für alle Werte ungleich 0 ein `true`:

```
$recht_schreiben = 0b10;
```

```
$recht_lesen     = 0b01;
```

```
$user_rechte     = 0b01;
```

```
// Diese Operation ergibt 01 und wird nach true konvertiert
```

```
if (true == (bool) ($user_rechte & $recht_lesen))
```

```
{
```

```
    echo "Darf lesen";
```

```
}
```

```
// Diese Operation ergibt 00 und wird nach false konvertiert
```

```
if (true == (bool) ($user_rechte & $recht_schreiben))
```

```
{
```

```
    echo "Darf schreiben";
```

```
}
```

Dieses kleine Beispiel gibt `Darf lesen` aus. Auf diesem Weg kann man eine ganze Menge interessanter Spielereien basteln. Beachten Sie aber immer, dass eine solche Implementation schon etwas speziell und nicht für jeden verständlich ist.

Insbesondere möchte ich noch darauf hinweisen, dass »Zahlen« wie Telefonnummern oder Postleitzahlen auf keinen Fall als Integer gespeichert werden sollten, da sie oft mit einer Null beginnen. Auch wenn das Wort Postleitzahl vermuten lässt, dass es sich um eine Zahl handelt, so sollten Sie solche Werte immer als String ablegen. Legen Sie nur dann einen Wert als Zahl ab, wenn Sie damit rechnen wollen.

### 2.1.3 Float

Float-Werte sind Fließkommazahlen, sie unterstützen also einen Nachkommanteil. Eine Unterscheidung zwischen Fließkommazahlen mit einfacher und doppelter Genauigkeit ist in PHP nicht vorgesehen. Auch wenn dieser Datentyp in PHP als `float` bezeichnet wird, so basiert er doch auch auf dem Datentyp `double` des verwendeten C-Compilers. Der verfügbare Wertebereich entspricht somit

auch dem, den das Betriebssystem bzw. der C-Compiler für `double` unterstützt. Typischerweise ist das der Bereich von  $1,8e-308$  bis  $1,8e+308$ . Die Genauigkeit beträgt hierbei 14 Stellen. Auch wenn dieser Wertebereich schon recht groß ist, so kann es natürlich passieren, dass das Ergebnis einer Berechnung den gültigen Fließkomma-Wertebereich verlässt. Um dies prüfen zu können, wurden mit PHP 4.2.0 die Funktionen `is_finite()` und `is_infinite()` eingeführt. Die erste liefert ein `true` zurück, wenn die übergebene Fließkommazahl ein endlicher Wert ist. Wurde einer Fließkommavariablen ein Wert zugewiesen, der größer ist als der zulässige Wertebereich, gibt sie ein `false` zurück.

```
$ok = 1e + 200;
$zu_gross = $ok * $ok;

if (true == is_finite($ok))
{
    echo "Der Wert ist OK";//Wird ausgegeben
}

if (true == is_infinite($zu_gross))
{
    echo "Der Wert ist zu groß";//Wird auch ausgegeben
}
```

Vergleicht man die hier beschriebenen Eigenschaften der Datentypen Integer und Float, könnte man auf die Idee kommen, immer nur mit Fließkommazahlen zu arbeiten. Leider sind diese aber sehr ungenau, so dass man sie wenn möglich vermeiden sollte.

```
$float = 0.3;
$float = $float + 0.3;
$float = $float + 0.3;
$float = $float + 0.1;
echo '$float: '.$float. '<br />';
echo 'floor() liefert: '.floor($float);
```

**Listing 2.1** Ungenauigkeit bei Fließkommaoperationen

So generiert dieses kleine Programm nicht, wie erwartet, 1 und 1 als Ausgabe, sondern:

```
$float:1
floor() liefert:0
```

Dieses Verhalten ist üblicherweise auch recht einfach mit einer `for`-Schleife nachzuweisen. So rechnet diese Schleife grundsätzlich korrekt:

# Index

`$_REQUEST[]` 650  
`$this` 163  
`.htaccess` 249  
`->` 158  
`@` 251  
`@abstract` 476  
`@access` 476  
`@author` 476  
`@copyright` 476  
`@deprecated` 476  
`@example` 476  
`@filesource` 476  
`@final` 476  
`@global` 477  
`@ignore` 478  
`@internal` 478  
`@license` 478  
`@link` 478  
`@name` 477  
`@package` 478  
`@param` 479  
`@return` 479  
`@see` 480  
`@since` 480  
`@static` 480  
`@staticvar` 481  
`@subpackage` 478  
`@todo` 481  
`@uses` 480, 481  
`@var` 481  
`@version` 481  
`__()` 751  
`__autoload()` 191  
`__call()` 180  
`__callStatic()` 183  
`__get()` 186  
`__NAMESPACE__` 204  
`__PHP_Incomplete_Class` 159  
`__set()` 186  
`__sleep()` 190  
`__toString()` 188  
`__wakeup()` 190  
404-Fehler 283

## A

---

Abkürzungen 109  
Ablaufgeschwindigkeit 757  
abstract 193  
Abstrakte Klassen 193  
accesskey 535  
Access-Keys 535  
Accordion 395  
Action Controller 334  
Active Recordset 351  
Airbus 391  
AJAX 548, 561  
alnum 574  
alpha 574  
Altersüberprüfung 696  
Anforderungsprofil (Pflichtenheft) 398  
Anker in regulären Ausdrücken 576  
Annahmen überprüfen 437  
apd 763  
Arithmetisches Mittel 146  
Array 68  
    *assoziatives* 68  
    *indiziertes* 68  
    *Loch in* 69  
    *sortieren* 79  
    *Suche in* 78  
    *zusammenführen* 73  
`array_diff()` 485  
`array_intersect()` 485  
`array_key_exists()` 79  
`array_map()` 76  
`array_merge()` 73, 485  
`array_pop()` 488  
`array_push()` 487, 488  
`array_reduce()` 75, 77  
`array_search()` 78  
`array_shift()` 487  
`array_unique()` 485  
`array_walk()` 75  
`array_walk_recursive()` 76  
`arsort()` 80  
ascii, POSIX-Klasse 574  
ASCII-Code 63  
ASCII-Code in regulären Ausdrücken 572

ASCII-Dateien 590  
 asort 80  
 ass\_call 440  
 Assembler 757  
 assert() 437  
 ASSERT\_ACTIVE 439  
 ASSERT\_BAIL 440  
 ASSERT\_CALLBACK 439, 440  
 assert\_options() 439  
 ASSERT\_QUIET\_EVAL 439  
 ASSERT\_WARNING 439  
 assertElementPresent() 427  
 Assertion 439  
 assertTable() 427  
 assertText() 427  
 assertTextPresent () 427  
 assertTitle() 427  
 assertValue() 427  
 assign (Smarty) 306  
 Ausgangsvoraussetzung  
   (Pflichtenheft) 397  
 Auskommentieren 97  
 Autocommit 806  
 autoload() 191

## B

---

Backreferences 578  
 Backslash 63  
 Balsamiq 394  
 Base64 614  
 base64\_decode() 744  
 base64\_encode() 743  
 Basisklasse 158  
 Bäume 493  
 BBCode 658  
 bcadd() 794  
 Bcc 611  
 bccomp() 795  
 bcdiv() 794  
 BCMath 793  
 bcmath() 794  
 bcmul() 794  
 bcpow() 794  
 bcpowmod() 795  
 bcsqrt() 795  
 bcsb() 794  
 Bedingte Ausdrücke 581

Bedingungen 115  
   *komplexe* 107  
   *Type Casting* 118  
 Benchmark 763  
 Benutzer-DSN 814  
 Betriebskonzept 398  
 Bibliotheken  
   *Abhängigkeiten* 153  
   *bedingte Funktionen* 154  
   *Dateiendung* 145  
   *Datenbankverbindung* 147  
   *externe* 143  
   *Fehlerbehandlung* 151  
   *Funktionen* 146  
   *Funktionsdesign* 149  
   *Nebeneffekte von Funktionen* 153  
   *Rückgabewerte von Funktionen* 151  
   *Tippfehler* 148  
   *Variablenfunktionen* 155  
   *veraltete Funktionen* 153  
   *Vorgabewerte für Parameter* 150  
   *Wiederaufrufbarkeit* 153  
   *Wrapper-Funktionen* 153  
 blank, POSIX-Klasse 574  
 Boolean 53  
 BOOLEAN MODE 819  
 Bootstrap File 331  
 Boundary 615  
 BPMN 393  
 break 127  
 Breakpoint 449

## C

---

Cache  
   *Cache\_Function()* 788  
   *Cache\_Output()* 789  
   *call()* 788  
   *end()* 790  
   *Garbage Collection* 791  
   *get()* 786  
   *isCached()* 786  
   *isExpired()* 786  
   *remove()* 787  
 Cache\_Function 788  
 cache\_lifetime (Smarty) 324  
 Cache-Systeme 784  
 Caching (Smarty) 323  
 cal\_days\_in\_month() 545

call() 180  
 Callback-Funktion bei assert 439  
 Call-Stack 450  
 callStatic() 183  
 capitalize 309  
 CAPTCHA 684  
   *reCAPTCHA* 692  
 Carriage Return 591  
 case-sensitive 132  
 Casting  
   *explizites* 66  
 catch 278  
 Cc 611  
 cd 590  
 CHAR 801  
 checkdate() 545  
 Check-In 45  
 chunk\_split() 617  
 class\_exists() 145, 207  
 CLF 285  
 Click-Event 552  
 Client URL 732  
 CMS 300  
 cntrl, POSIX-Klasse 574  
 Coder 401  
 Code-Review 400  
 Codesniffer 402  
 Coding Standard 402  
 Comma-separated Values 595  
 Commit 805  
 Common Logfile Format → CLF  
 Consolidate Duplicate Conditional  
   Fragments 138  
 Content-Management-System → CMS  
 continue 127  
 Controller 401  
 Copy and Paste Detector 405  
 Copy-on-Write 759  
 Copyright 475  
 CouchDB  
   *POST* 858  
   *View* 859  
 couchDB  
   *emit()* 860  
 count\_characters 310  
 count\_paragraphs 310  
 count\_sentences 310  
 count\_words 310  
 crack\_opendict() 678

CrackLib 677  
 CREATE TABLE 734  
 Cross-Site Request Forgeries → CSRF  
 Cross-Site Scripting → XSS  
 CSRF 659  
 CSV-Dateien 595  
 cURL 732  
 curl\_exec() 732  
 curl\_init() 732  
 curl\_setopt() 732  
 CVS 41

## D

---

Data Source Name 814  
 date() 739  
 date\_format (Smarty) 310  
 Datei-DSN 814  
 Dateien 588  
   *unterschiedlicher Betriebssysteme* 590  
 Dateilock 599  
 Dateirechte 589  
 Datenbank 795  
   *Datenformate* 801  
   *INTEGER* 802  
 Datenbankabfragen  
   *Performance* 779  
 Datenformat  
   *Datenbanken* 801  
 Datentypen 53  
 Datumsformat 742, 752  
 daylight 740  
 debug 591  
 debug (Smarty) 323  
 debug\_backtrace() 437, 442  
 debug\_print\_backtrace() 437, 442  
 Debug-Features 437  
 Debugger  
   *professionelle* 447  
 Debugging 437  
   *lokales* 447  
   *Werte ändern* 451  
 Debug-Routine  
   *eigene* 442  
 Debug-Sessions 448  
 decimal\_point 755  
 Decompose Conditional 136  
 default\_handler() 256  
 Dekrement 126



Delimiter 571  
 Denormalisierung 830  
 Design Pattern 221  
 digit, POSIX-Klasse 574  
 DIN 66001 465  
 DIN 66230 460  
 DIN 66231 460  
 DIN 66232 460  
 Directory 159, 589  
 disk\_free\_space() 725  
 Diskussionsforum 501  
 display 306  
 display\_errors 249  
 dividiere() 272  
 DocBlock 473  
 DocBlocks 24  
 Dokumentation 459  
   *Abhängigkeiten* 462  
   *Anforderungen* 460  
   *Aufgabenstellung* 460  
   *Ausgangsvoraussetzung* 460  
   *globale Datenstrukturen* 463  
   *globale Variablen* 462  
   *Klassen* 462  
   *Konstanten* 462  
   *Leistungsbeschreibung* 462  
   *Maßeinheiten* 461  
   *Namenskonventionen* 461  
   *Schnittstellen* 461  
   *Style-Guide* 461  
   *Topdown-Strategie* 461  
   *Versionierung* 461  
   *Versionsnummer* 462  
   *Verzeichniskonventionen* 461  
   *Zugriffsmodifikatoren* 463  
   *Zweck der Datei* 462  
 DOM 428  
 Double Ticks 61  
 DROP TABLE 736  
 DSN 814

## E

---

E\_ERROR 247  
 E\_NOTICE 247  
 E\_USER\_ERROR 247  
 E\_USER\_NOTICE 247  
 E\_USER\_WARNING 247  
 E\_WARNING 247  
 Eclipse-PDT 19  
 Editor  
   *PHPedit* 18  
   *PhpStorm* 19  
 eh() 264  
 Eigenschaften 157, 160  
 Einrückung 105  
 Einzelstückmuster 221  
 ELF 285  
 Elternklasse 158  
 Emacs 592  
 E-Mail-Adressen  
   *Schutz* 702  
 E-Mails 605  
   *Absender* 609  
   *Base64* 614  
   *base64\_encode()* 616  
   *Bcc* 611  
   *Boundary* 615  
   *Cc* 611  
   *chunk\_split()* 617  
   *Content-Transfer-Encoding* 613  
   *Dateianhang* 615  
   *eingebundene Grafiken* 620  
   *Empfänger* 607  
   *Empfangsbestätigung* 611  
   *Envelope* 605, 610  
   *Errors-To* 610  
   *first-class* 613  
   *GnuPG* 625  
   *Header* 605  
   *HTML* 615  
   *komprimieren* 623  
   *Kopien* 611  
   *Lesebestätigung* 611  
   *MIME* 613  
   *PGP* 625  
   *Precedence* 613  
   *Priorität* 612  
   *proc\_open()* 629  
   *Return-Receipt-To* 612  
   *RFC 1342* 614  
   *RFC 2822* 606  
   *str\_replace()* 614  
   *text/plain* 613  
   *Troubleshooting* 609  
   *Umlaute* 613  
   *Umlaute in Betreffzeile* 614  
   *urlencode()* 614

- E-Mails (Forts.)
    - verschlüsseln* 625
    - X-Priority* 612
    - Zeichensatz* 606
    - Zeilenumbrüche* 607
  - Embedded PHP 88
  - emit() 860
  - Empfangsbestätigung 611
  - empty() 131
  - Enclosure 597
  - Entwicklungsserver 32, 452
  - Entwurfsmuster 221
  - Envelope 605, 610
  - Ereignisorientierung 552
  - ERROR 246
  - Error 404 283
  - Error Handler 246
  - Error Handling 245
    - eigene Error Handler* 252
    - in Bibliotheken* 268
    - in Klassenbibliotheken* 273
    - kundenfreundliche Fehlermeldung* 259
    - Logfile* 260
    - Negativbeispiel* 246
    - send\_mail()* 264
    - SMS* 264
  - Error Tracking 251
  - error\_log 249
  - error\_reporting 247
  - escape (Smarty) 312
  - escapeshellarg() 673
  - Event Handling 552
  - Excel 597
  - Exception Handling in PHP 5 278
  - Exception werfen 279
  - exp 735
  - Extended Logfile Format → ELF
  - extends 169
  - Externe Bibliotheken 143
  - Extract Method 136
  - extract() 650
  - eXtreme Programming → XP
- F**
- 
- Fabrikmuster 227
  - Factory-Pattern 227
  - Fakultät 495
  - Fallstricke 100
  - Fallunterscheidung 120
  - false 53
  - Farbenblindheit 533
  - fclose() 101
  - Fehlerbehandlung 245
  - Fehlerdokumente 283
  - Fehler-Kontrolloperator 251
  - Fehlermanagement 259
  - fgetcsv() 595
  - fieldset 534
  - final 201
  - Finale Klassen 201
  - FIXME 101
  - Fließkommazahl 56
  - Float 53, 56
  - flock() 600, 781
  - foreach 71, 127
  - Formatierung 102
  - Formular
    - Aufbau* 533
    - clientseitige Prüfung* 544
    - Default-Werte* 535
    - Feldbreite* 536
    - Gliederung* 536
    - isset* 538
    - Leserichtung* 537
    - Optionsliste* 535
    - Plausibilitätskontrolle* 544
    - Postleitzahlen prüfen* 546
    - Select-Box* 535
    - Submit-Button* 537
    - Validitätsprüfung* 544
    - Value-Konvertierung* 538
    - variable Feldanzahl* 540
    - Wertübergabe mit Arrays* 540
    - Wertübernahme* 537
  - frac\_digits 755
  - Frameworks 325
    - Außendarstellung* 299
    - Auswahl von* 292
    - Bugs* 295
    - Codequalität* 294
    - Dokumentation* 296
    - Funktionsumfang* 293
    - Lizenz* 298
    - Performance* 297
    - Spezielle Features* 298
    - Support* 297
  - freshmeat.net 291

## Index

Front Controller 333  
ftruncate() 602  
function\_exists() 145, 724  
Funktionsaufruf  
    *verschachtelter* 105  
Funktionsnamen 113

## G

---

Genau rechnen 792  
Genauigkeit von Fließkommazahlen 56  
generateID() 785  
Generation 719  
get() 186  
get\_class() 209  
get\_class\_methods() 208  
get\_class\_vars() 208  
get\_declared\_classes() 207  
get\_object\_vars() 211  
get\_parent\_class() 211  
getCode() 279  
getcwd() 725  
getFile() 279  
getLine() 279  
getMessage() 281  
getPost() 341  
getText() 749  
Gier 582  
Git 41  
Globale Variable 102  
gmdate() 739  
GMP 794  
GMT 737  
gmtime() 740  
GnuPG 625  
Grafiken in Installationsprogrammen 743  
graph, POSIX-Klasse 574  
greedy 582  
Greenwich Mean Time → GMT  
Groß-/Kleinschreibung 132

## H

---

Haltepunkt 449  
Harvester 702  
Header 598, 605  
Helios 27  
Helper 350

heredoc 62  
Hexadezimale Zahl 54  
HexEdit 592  
Hinting 192  
hotscripts.com 291  
htaccess 146  
    *display\_errors* 250  
    *error\_log* 250  
    *error\_reporting* 250  
    *log\_errors* 250  
http-Code 565  
HTTP-Statuscode 287

## I

---

i18n 749  
if (Smarty) 317  
IIS 332  
imagecolorallocate() 704  
imagecreate() 704  
imagettfbbox() 704  
imagettftext() 704  
implements 195  
in\_array() 78, 485  
include 102  
include (Smarty) 322  
include\_once() 143  
Info-ZIP 623  
ini\_set() 250  
Inkrement 126  
InnoDB 806  
Input-Felder 537  
Installation von Komponenten 730  
Installationsprogramme 718  
    *Grafiken* 743  
Installationsvoraussetzungen 718  
instanceof 192  
INTEGER  
    *Datenbanken* 802  
Integer 53, 54  
    *Überlauf* 65  
Interaktion mit Benutzern 533  
Interceptor-Methoden 180  
Interface 193  
Internationalisierung 749  
inter-process communication remove 718  
inter-process communication status 717  
Interprozess-Kommunikation 706

Introspektion 206  
 IPC 706  
 ipcrm 717  
 ipcs 717  
 is\_a() 210  
 is\_cached (Smarty) 324  
 is\_readable() 725  
 is\_subclass\_of() 211  
 is\_writable() 725  
 ISBN 548  
 isCached() 786  
 isError() 274  
 isExpired() 786  
 ISO 3166 753  
 ISO 639-1 753  
 isset 538  
 isset() 187

## J

---

Java 278  
 JavaScript 549  
   *Ereignisse* 552  
   *Variablen* 550  
 JavaScript in PHP 106  
 JDBC 28  
 JMeter 429  
 jQuery 549  
   *ajax()* 562  
   *Fehlermeldungen* 561  
   *Formular mit* 557  
   *messages* 561  
   *parseJSON()* 566  
   *Plug-ins* 557  
   *rules* 559  
   *Validierung* 557  
 JSON 555  
 json\_encode 565

## K

---

Kaffee kochen 388  
 Kapselung 164  
 Kindklasse 158  
 Klammerung 103  
 Klasse 157  
   *Deklaration* 159  
   *untersuchen* 207

Klassennamen 113  
 Klick-Event 552  
 Klonen von Objekten 185  
 Knoten 493  
 Kommentar 96  
   *mehrzeiliger* 97  
 Komodo 447  
 Komponenten  
   *Installation* 730  
 Konfigurationsdateien 727  
 Konstruktor 161  
 Kontonummern 546  
 Kontrollstrukturen 115  
 Konzeption 388  
 Kreditkartennummern 548  
 krsort() 80  
 ksort() 80

## L

---

l10n 749  
 LAMP 32  
 Lasttests 429  
 Laufbedingung 118  
 LC\_ALL 752  
 legend 534  
 Lesebestätigung 611  
 libcurl 732  
 Lieferumfang (Pflichtenheft) 398  
 LiFo 488  
 Line Feed 591  
 linksassoziativ 115  
 literal (Smarty) 316  
 localeconv() 755  
 Lock 599  
 LOCK\_EX 600  
 LOCK\_UN 601  
 Locking-Mechanismus 602  
 log\_errors 249  
 Logische Fehler 437  
 Lokales Debugging 447  
 Lokalisierung 749  
 Lookahead 578  
 Lookaround 578  
 Lookbehind 578  
 lower, POSIX-Klasse 575  
 ls 590  
 Lufthansa 391

## M

---

- Magic Numbers 99
- Magic Quotes 804
- magic\_quotes\_gpc 665, 804
- magic\_quotes\_runtime 804
- Magische Methoden 180
- mail() 605
- mailto 705
- MAMP 33
- Maschinen-Code 757
- Mehrsprachige Texte 749
- Member-Funktionen 157
- Member-Methode 113
- Member-Variablen 157
- Mengen 485
  - Array 485
- Mengengerüst (Pflichtenheft) 398
- Mess Detector 406
- Messenger 709
- method\_exists() 210
- Methode
  - private 113
- Methoden 157, 161
- MIME 613
- Mini-Blog 378
- Mitwirkungspflicht des Kunden
  - (Pflichtenheft) 398
- Mock-ups 393
- mod\_rewrite 333
- Model-View-Controller 325
- mon\_decimal\_point 755
- mon\_thousands\_sep 755
- money\_format() 753
- Muster 221
- MVC 325
  - \_\_call() 344
  - Action Controller 334
  - assign() 339
  - canSendHeaders() 349
  - Controller-Benennung 336
  - Datenübergabe 339
  - dispatch() 334
  - Error Handling 344
  - errorAction 347
  - ErrorController 347
- MVC (Forts.)
  - escape() 340
  - Exception Handling 334
  - Exceptions 334
  - forward() 345
  - Front Controller 333
  - getParam() 342
  - getPost() 341
  - getRequest() 341
  - getResponse() 348
  - getStaticHelper() 350
  - GET-Werte 341
  - Header 349
  - init() 349
  - mehrere Controller 336
  - Model 350
  - Moved Permanently 346
  - Moved Temporarily 346
  - Parameter 342
  - POST-Werte 341
  - redirect() 346
  - Request-Objekt 341
  - setControllerDirectory() 334, 335
  - setParam() 346
  - throwExceptions 334
  - throwExceptions() 346
  - Übergabe von Werten 341
  - useDefaultControllerAlways 344
  - Verzeichnisstruktur 331
  - View 337
  - View unterdrücken 334
  - ViewRenderer 350
- myError 274
- MyISAM 806
- MySQL
  - Release Level 723
  - Version 722
- mysql\_free\_result() 101
- mysql\_get\_server\_info 722
- mysqldump 735
- mysqli 806
  - mysqli\_autocommit() 806
  - mysqli\_commit() 807
  - mysqli\_query 807
  - mysqli\_rollback 807
  - mysqli\_select\_db 807
- MySQL-ODBC 814

**N**

---

Nachrichten-ID 713  
 Namen  
   *für Funktionen* 113  
   *für Klassen* 113  
   *für Konstanten* 110  
   *für Variablen* 110  
 Namensgebung 108  
 Namensräume 202  
 Namespaces 202  
 Nassi-Shneidermann-Diagramm → NSD  
 Natural Key 354  
 new 158  
 Nicht deklarierte Eigenschaften 186  
 Nicht deklarierte Methoden 180  
 nl2br (Smarty) 313  
 nl2br() 594  
 Normalisierung 796  
 Notepad 17  
 NOTICE 246  
 nowdoc 62  
 NSD 465  
 NTP 740  
 NULL 85  
 number\_format() 753  
 NuSphere 18

**O**

---

Oberflächentests 423  
 Objekte 157  
   *Case-Sensitivity* 159  
   *Deklaration von Eigenschaften* 160  
   *Deklaration von Klassen* 159  
   *Schnittstellen* 163  
 Objektorientierte Programmierung  
   *deklarierte Klassen auslesen* 207  
   *Introspektion* 206  
   *PHP-interne Klassen* 208  
   *Untersuchen von Objekten* 209  
 Objektorientierte Programmierung →  
 OOP  
 Objekt-relationales Mapping → ORM  
 Observable 236  
 Observer-Pattern 236  
 ODBC 813  
 odbccommit() 818  
 odbccconnect() 816

odbc\_error() 817  
 odbcerrormsg() 817  
 odbccexec() 816  
 odbccfetch\_row() 817  
 odbccresult() 817  
 odbccrollback() 818  
 Oktale Zahl 54  
 onclick 553  
 OOP 157  
 Open Database Connectivity 813  
 Operator  
    *Casting-Operatoren* 66  
    *ternärer* 123  
    *trinärer* 123  
    *Typ des Rückgabewerts* 64  
    *Verknüpfungsoperator* 61  
 Optionen für PCREs 582  
 Ordnung 101  
 ORM 327  
 Output\_Cache 789  
 Overhead 157

**P**

---

Pair Programming 401  
 PAP 463  
 Parameter 133  
 parent 170  
 parse\_ini\_file() 728  
 Passphrasen 677  
 Passwort 675  
    *generieren* 680  
    *neues* 676  
    *speichern* 683  
 Pattern 221  
 PCRE 570  
    *Delimiter* 571  
    *Optionen* 582  
    *Quantifizier* 575  
 PCS 139  
 PDT 19  
    *Konfiguration* 29  
    *SVN* 43  
 PEAR-Coding-Standard → PCS  
 PEAR-Installer 51  
 Peer-Review 400  
 Pencil 394  
 Performance vs. Lesbarkeit 94  
 Performance-Bremsen 762

- Performance-Tuning 757
- Perl Compatible Regular Expression →
  - PCRE
- Pflichtenheft 397
- pg\_dump 735
- PGP 625
- PGPDesktop 627
- PHP
  - neue Versionen* 134
- PHP Codesniffer 402
- PHP Error Log 36
- PHP Mess Detector → PHPMD
- php.ini
  - display\_errors* 249
  - error\_log* 249
  - error\_reporting* 247
  - log\_errors* 249
- php\_version() 720
- phpcpd 405
- phpcs 402
- phpDocumentor 468
  - CHM* 483
  - globale Variablen* 477
  - HTML-Darstellung* 483
  - Packages* 478
  - PDF* 483
  - Subpackages* 478
  - Tag* 475
- PHPEdit 18
- php-free.de 291
- phpinfo() 34
- PHP-Land-Caches 784
- PHPMD 406
- PHPMyAdmin 807
- PhpStorm 19
- PHP-Syntax
  - alternative* 95
- PHPUnit 412
- PHPUnit\_Framework\_TestCase 414
- Plausibilitätskontrolle 544
- Polymorphie 201
- Port 37 740
- POSIX-Standard 574
- PostIdent 696
- Postinkrement 757
- precision (php.ini) 59
- preg\_grep() 587
- preg\_match() 571, 584
- preg\_match\_all() 585

- preg\_replace() 586
- preg\_replace\_callback() 587
- preg\_split() 588
- Preinkrement 759
- Primzahl
  - berechnen* 760
- print, POSIX-Klasse 575
- private 171
- Private Key 625
- proc\_open() 629
- Produktivserver 452
- Professionelle Debugger 447
- Programmablaufplan 463
- Programmierstil 87
- Projekte
  - Dokumentation von* 459
- protected 171
- Prototypen 157, 398
- Proxy-Pattern 232
- Prüfsummenalgorithmen 546
- Prüfziffer 547
- public 171
- Public Key 625
- punct, POSIX-Klasse 575

## Q

---

- Qualifier 112
- Qualitätsmerkmale 399
- Qualitätssicherung 387
- Quantifier 575
- Quelltext-Formatierung 102
- Query-Caching 779
- Queue 487

## R

---

- Race Condition 781
- raiseError() 274
- Ramp-up Period 432
- RC1 719
- reCAPTCHA 692
- Refactoring 135
  - Consolidate Duplicate Conditional Fragments* 138
  - Extract Method* 136
  - Rename Method* 135
- Reflection-API 212
- ReflectionClass 213

ReflectionFunction 219  
 ReflectionMethod 217  
 ReflectionParameter 218  
 ReflectionProperty 218  
 Reflexion 206  
 RegEx 570  
 regex\_replace (Smarty) 314  
 register\_globals 645  
 register\_shutdown\_function() 714  
 Reguläre Ausdrücke 570  
     *ASCII-Code* 572  
     *Kommentare* 576  
 Reihenfolge von Datenbankeinträgen 801  
 Rekursion 493  
 Release Candidate 719  
 Remote Debugging 452  
 Rename Method 135  
 require 102  
 require\_once() 143  
 Resource 86  
 restore\_error\_handler() 255  
 Return-Receipt-To 612  
 Reviews 400  
 Rewrite-Engine 332  
 Rewrite-Rule 332  
 RFC 1305 740  
 RFC 1342 614  
 RFC 2030 740  
 RFC 2045 617  
 RFC 2445 740  
 RFC 2822 605  
 RFC 868 740  
 Rot-Grün-Schwäche 533  
 rsort() 80  
 RTFM 16  
 Rückwärtsreferenz 578  
 Ruleset 407

## S

---

Schleife 125  
     *typische Einsatzbereiche* 127  
 section (Smarty) 318  
 Selenese 426  
 Selenium 423  
     *DOM* 428  
 Selenium IDE 423  
 sem\_acquire() 708  
 sem\_get() 708  
 sem\_release() 708  
 sem\_remove() 708  
 Semaphor 707  
 Serialisieren von Objekten 190  
 Server 32  
 Serverzeit 736  
 session\_destroy() 637  
 session\_set\_cookie\_params() 638  
 session\_set\_save\_handler() 640  
 session\_start() 637  
 set() 186  
 set\_error\_handler() 253  
 setlocale() 752  
 Shared Memory 706  
 Shared Server 134  
 shed 592  
 Shell-Injections 671  
 shm\_attach() 707  
 shm\_detach() 708  
 shm\_get\_var() 708  
 shm\_put\_var() 708  
 shm\_remove() 708  
 ShoppingCart 158  
 show tables 748  
 Sicherheit 632  
     *Altersüberprüfung* 696  
     *BBCode* 658  
     *Bugfixes* 635  
     *CAPTCHA* 684  
     *CrackLib* 677  
     *CSRF* 659  
     *E-Mail-Adressen* 702  
     *escapshellarg()* 673  
     *fertige Lösungen* 634  
     *Geltungsbereiche* 649  
     *Generieren von Passwörtern* 680  
     *Globals* 645  
     *Harvester* 702  
     *htmlentities()* 657  
     *neue Passwörter* 676  
     *Passphrasen* 677  
     *Passwort* 675  
     *register\_globals* 645  
     *Session-ID* 637  
     *Sessions* 636  
     *Shell-Injections* 671  
     *Social-Engineering* 676  
     *Speichern von Passwörtern* 683  
     *SQL-Injection* 663



## Index

- strip\_tags()* 657
  - unset()* 648
  - variable Funktionen* 674
  - Sicherheit (Forts.)
    - Variablen initialisieren* 647
    - Vulnerability* 632
    - XSS* 654
    - Zugriffsrechte* 667
  - Simonyi, Charles 111
  - Singleton 221
  - Skalare Datentypen 53
  - sleep()* 190, 601
  - Smarty 300
    - Arrays nutzen* 307
    - assign* 306
    - cache\_lifetime* 324
    - Caching* 323
    - capitalize* 309
    - count\_characters* 310
    - count\_paragraphs* 310
    - count\_sentences* 310
    - count\_words* 310
    - date\_format* 310
    - debug* 323
    - display* 306
    - einbinden* 304
    - escape* 312
    - Funktionen* 316
    - if* 317
    - include* 322
    - Installation* 301
    - is\_cached()* 324, 325
    - literal* 316
    - Modifikatoren* 308
    - nl2br* 313
    - regex\_replace* 314
    - section* 318
    - spacify* 309
    - strip\_tags* 314
    - superglobale Variablen* 308
    - truncate* 315
    - upper* 309
    - wordwrap* 315
  - SNTP 740
  - Social-Engineering-Angriffe 676
  - Softwareumgebung 719
  - Sommerzeit 738
  - sort()* 79
  - space, POSIX-Klasse 575
  - spacify 309
  - Späte Bindung 443
  - Speicherplatz, verfügbarer 725
  - Speichersegment 707
  - spl\_autoload\_register()* 206
  - SQL in PHP 106
  - SQL-Injection 663
  - Stack 488
  - Stapel 488
  - static 176
  - Statische Seite 775
  - stdClass* 159
  - Stellvertreter-Muster 232
  - Step Into 451
  - Step Over 451
  - Step Return 451
  - strftime()* 753
  - String 53, 60
    - maximale Länge* 60
    - Zugriff auf einzelne Zeichen* 63
  - strip\_tags* (Smarty) 314
  - stripslashes()* 805
  - Struktogramm 463
  - Strukturmuster 232
  - Subklasse 158
  - Submit-Button 537
  - Submuster 577
  - Subversion 40
  - sum()* 802
  - Superklasse 158
  - SVN 40
    - Check-In* 45
    - Client* 43
    - Konflikt* 48
    - PDT* 43
    - Port* 42
    - Revert* 49
    - Zugriff* 44
  - System-DSN 814
  - sysvshm.init\_mem* 708
- ## T
- 
- Tabellenstruktur in phpMyAdmin 810
  - tabindex* 534
  - Tabulatorsprung 63
  - Team-Review 400
  - Template-Engine 300
  - Templates 300

Ternärer Operator 123  
 Testing 192  
 Testplan 431  
 Textareas 537  
 TextEdit 17  
 thousands\_sep 755  
 throw 279  
 Ticks 61  
 Time Protocol 740  
 TIMESTAMP  
   MySQL 803  
 ToDo 100  
 Token 776  
 TortoiseSVN 43  
 toString() 188  
 track\_errors 251  
 Traits 197  
 Transaktionsorientierung 805  
 TRICKY 101  
 trigger\_error() 257  
 Trinärer Operator 123  
 true 53  
 truncate (Smarty) 315  
 try 278  
 Turck MMCache 784  
 Type Casting 53  
   *automatisches* 64  
 Type Hinting 192  
 Type Testing 192  
 Typkonvertierung 64

## U

---

uasort() 80  
 uksort() 80  
 UltraEdit 592  
 Underscore 113  
 Ungarische Notation 111  
 Uninstall 746  
 Uninstall-Informationen 746  
 Unit 437  
 UNIX System V 707  
 UNIX-Dateisystem 589  
 unset() 101, 187  
 Untersuchen von Objekten 209  
 UPDATE 803  
 upper 309  
 upper, POSIX-Klasse 575  
 URL-Rewriting 332

use 205  
 Use-at-Will 329  
 useDefaultControllerAlways 344  
 Userland-Caches 784  
 usort() 80  
 USS Vincennes 393  
 USS Yorktown 392

## V

---

Validitätsprüfung 544  
 Value-Konvertierung 538  
 VARCHAR 801  
 Variable  
   *globale* 102  
   *Qualifier* 112  
 Variable Funktionen 674  
 Variablendeklaration 53  
 Variablenfunktionen 155  
 Variablenname 108  
 Varying Character 801  
 VBA 597  
 Vererbung 158, 167  
 Verhaltensmuster 236  
 verifyTextPresent 426  
 Verkettete Liste 488  
 Versionskontrolle 40  
 Versionsnummer 475, 721  
 Verständlichkeit des Codes 93  
 Verzeichnisstruktur 331, 724  
 Vielgestaltigkeit 201  
 View 337  
   *Speicherort* 338  
 Visual Basic for Applications → VBA  
 Volltextsuche 818  
 Vorausschauen 578  
 Vorletzter Fehler 387  
 Vulnerability 632

## W

---

Wagenrücklauf 63  
 wakeup() 190  
 WAMP 32  
 WARNING 246  
 Warschau 391  
 Warteschlange 487  
 Watchdog 412  
 Whitespace 63

wordwrap (Smarty) 315  
 Workaround 100  
 Wrapper-Funktionen 153  
 Wurzel 493

## X

---

XAMPP 33  
 Xdebug 763  
 xdigit, POSIX-Klasse 575  
 XHR 562  
 XMLHttpRequest 562  
 XP 401  
 X-Priority 612  
 XSS 654  
 XX\_DEBUG\_ON 443

## Z

---

Zahlenformat 752  
 Zeichenklassen 573  
 Zeichensatz 595  
 Zeilenumbruch 403  
 Zeilenvorschub 63  
 Zeitzone 736  
 Zend Data Cache 36  
 Zend Debugger 36  
 Zend Framework  
   *Error Handling* 344  
   *Front Controller* 333  
   *Mini-Blog* 378  
   *Nutzung* 329  
 Zend Optimizer+ 36  
 Zend Performance Suite 784  
 Zend Server 33  
   *Konfiguration* 36  
   *Logs* 35  
   *PHP Erweiterungen* 37  
 Zend Studio 31  
 Zend\_Controller\_Action 334  
   *getResponse()* 348

Zend\_Controller\_Action\_Exception 344  
 Zend\_Controller\_Dispatcher\_Exception 344  
 Zend\_Controller\_Front 331, 333  
   *getInstance()* 334  
   *setControllerDirectory()* 335  
   *setParam* 334  
 Zend\_Controller\_Request\_Http 341  
 Zend\_Db  
   *Optionen* 355  
   *unterstützte Datenbanken* 355  
 Zend\_Db\_Expr 358  
 Zend\_Db\_Table 351  
   *Einfügen von Daten* 358  
   *fetchAll()* 361  
   *fetchRow()* 362  
   *insert()* 358  
   *LIMIT* 362  
   *Löschen von Daten* 359  
   *Natural Key* 354  
   *Primärschlüssel* 353  
   *save()* 363  
   *SELECT* 360  
   *setFromArray()* 363  
   *setupTableName()* 352  
   *Tabellenname* 352  
   *update* 358  
 Zend\_Db\_Table\_Row 360  
 Zend\_Db\_Table\_Rowset 360  
 Zend\_Form 365  
   *Filter* 376  
   *Pflichtfelder* 372  
   *reguläre Ausdrücke* 375  
   *Validator* 372  
 Zend\_View 337, 339  
   *escape()* 340  
 Zend\_View\_Exception 337  
 Zugriffs-Logfiles 285  
 Zurückschauen 578