

stefan PRIEB SCH



# PHP MIGRIEREN

KONZEPTE UND LÖSUNGEN ZUR  
MIGRATION VON PHP-ANWENDUNGEN  
UND -UMGEBUNGEN



HANSER



# Inhalt

<b>1</b>	<b>Einführung.....</b>	<b>1</b>
1.1	Eine kurze Geschichte der Internet-Zeit.....	1
1.2	PHP erblickt das Licht der Welt.....	3
1.3	PHP 5 und die große Migration.....	5
1.4	Blick in die Zukunft: PHP 6.....	7
<b>2</b>	<b>Strategien.....</b>	<b>9</b>
2.1	Never touch a running system.....	10
2.1.1	Systemumgebung.....	11
2.1.2	Programmcode.....	13
2.2	Immer die neueste Version einsetzen.....	16
2.3	Auf der grünen Wiese beginnen.....	18
2.3.1	Programmcode von Grund auf neu entwickeln.....	19
2.3.2	Die Systemumgebung von Grund auf neu aufbauen.....	20
2.4	Den richtigen Mittelweg finden.....	21
<b>3</b>	<b>Aspekte einer Migration.....</b>	<b>23</b>
3.1	Plattform.....	25
3.1.1	Architektur.....	25
3.1.2	Prozessor.....	26
3.1.3	Instruction Set.....	27
3.1.4	Wortlänge.....	28
3.1.5	Byte-Reihenfolge.....	30
3.1.6	Bauformen und Schnittstellen.....	31
3.2	Betriebssystem.....	32
3.2.1	Wortlänge.....	35
3.2.2	Zeilenendezeichen.....	35
3.2.3	Zugriffsrechte.....	38
3.2.4	Pfade und Dateinamen.....	40
3.2.5	Temporäre Dateien.....	43

3.2.6	Der Dateisuchpfad .....	44
3.2.7	Zeichensätze .....	44
3.3	Datenbank .....	47
3.3.1	SQL ist nicht gleich SQL.....	48
3.3.2	Programmcode in der Datenbank .....	49
3.3.3	Datentypen.....	51
3.3.4	Zeichensätze .....	51
3.3.5	Backup und Restore.....	53
3.4	Webserver .....	56
3.4.1	Apache und Apache2.....	57
3.4.2	Sicherheit.....	58
3.4.3	Apache übersetzen .....	60
3.4.4	Mehrere Webserver auf einem Rechner .....	61
3.5	PHP .....	62
3.5.1	Integration mit dem Webserver .....	62
3.5.2	PHP übersetzen.....	67
3.5.3	Thread-Modell.....	68
3.5.4	Die PHP-Konfiguration .....	69
3.5.5	PHP-Erweiterungen .....	87
3.5.6	Mehrere PHP-Versionen installieren .....	94
3.6	PHP-Code.....	94
3.6.1	Code von Drittanbietern .....	95
3.6.2	Eigener PHP-Code.....	96
3.7	Externe Programme.....	97
3.8	Schnittstellen zu Drittsystemen .....	100
3.9	Zeichenkodierungen .....	101
3.10	Browser .....	109
3.11	Weitere Aspekte .....	111
3.11.1	Harte Limits.....	111
3.11.2	Sicherheit.....	113
3.11.3	Systemkonzept.....	114
<b>4</b>	<b>Die Migration vorbereiten .....</b>	<b>117</b>
4.1	Der Istzustand.....	118
4.2	Das Zielsystem.....	121
4.3	Die Migration planen .....	124
<b>5</b>	<b>Die Migration durchführen.....</b>	<b>129</b>
5.1	Vorbereitungen.....	131
5.1.1	Das erste Testsystem .....	131
5.1.2	Das zweite Testsystem.....	132
5.2	Testen .....	133
5.2.1	Relevante Testfälle finden .....	134
5.2.2	Testdaten finden .....	135
5.2.3	Tests programmieren .....	136

5.3	Refactoring.....	138
5.3.1	Redundanten Code eliminieren.....	139
5.3.2	Codeblöcke kürzen.....	140
5.3.3	Unterschiedliche Belange trennen.....	140
5.4	Migrieren.....	142
5.4.1	Altlasten beseitigen.....	143
5.4.2	Module ersetzen.....	144
5.4.3	Syntaxfehler beseitigen.....	145
5.4.4	Alle PHP-Fehlermeldungen beseitigen.....	145
5.4.5	Logische Fehler beseitigen.....	146
5.4.6	Die PHP-Konfiguration normieren.....	146
5.5	Das Live-System migrieren.....	147
5.6	Die Migration abschließen.....	150
<b>6</b>	<b>Nach der Migration.....</b>	<b>153</b>
6.1	Gesammelte Erfahrungen.....	153
6.1.1	Modular programmieren.....	154
6.1.2	Coding Guidelines etablieren.....	154
6.1.3	Defensiv programmieren.....	154
6.1.4	Nicht der Erste sein.....	155
6.1.5	Laufendes Refactoring.....	156
6.1.6	Agile Migration.....	157
<b>7</b>	<b>Werkzeuge.....</b>	<b>159</b>
7.1	Versionsverwaltung.....	160
7.2	Kommandozeilenwerkzeuge.....	161
7.2.1	HTTP-Anfragen senden und Dateien herunterladen: wget.....	161
7.2.2	Dateien und Verzeichnisse durchsuchen: grep.....	163
7.2.3	Ersetzungen in Dateien vornehmen: sed.....	164
7.2.4	Dateien und Verzeichnissen vergleichen: diff-Werkzeuge.....	164
7.3	(X)HTML-Dateien validieren.....	166
7.3.1	Der W3C-Validator.....	168
7.3.2	HTML Tidy.....	170
7.3.3	Die PHP-Erweiterung Tidy.....	172
7.4	CSS-Dateien validieren.....	174
7.5	XML-Dateien validieren.....	176
7.5.1	xmllint.....	176
7.5.2	PHP.....	178
7.6	Statische Analyse von JavaScript-Dateien.....	178
7.6.1	jshint.....	179
7.6.2	JSLint.....	180
7.7	Firefox-Erweiterungen.....	182
7.7.1	Webdeveloper.....	182
7.7.2	XPather.....	184
7.7.3	Firebug.....	185

7.8	PHP-Bordmittel.....	187
7.8.1	PHP-Konfiguration.....	187
7.8.2	Syntaxprüfung.....	190
7.8.3	Prepend- und Append-Dateien.....	191
7.9	PEAR-Komponenten.....	192
7.9.1	PHP_Compat.....	194
7.9.2	PHP_Beautifier.....	195
7.9.3	PHP_CodeSniffer.....	199
7.9.4	PHP_CompatInfo.....	205
7.10	Virtuelle Maschinen.....	207
7.10.1	VMWare.....	208
7.10.2	Eine virtuelle Maschine installieren.....	209
7.10.3	Mit Snapshots arbeiten.....	211
7.11	Testwerkzeuge.....	212
7.11.1	Komponententests mit PHPUnit.....	213
7.11.2	Systemtests mit Selenium.....	216
7.12	Programmanalyse und Debugging.....	225
7.12.1	Installation.....	225
7.12.2	Nützliche Features.....	226
7.12.3	Tracing.....	226
7.12.4	Debugging.....	228
7.12.5	Codeabdeckung.....	231
7.13	Build-Automation.....	232
7.13.1	Installation.....	232
7.13.2	Codequalitätssicherung.....	233
7.13.3	Testautomation.....	235
7.13.4	Codeabdeckungsstatistiken.....	237
<b>8</b>	<b>PHP-Code migrieren.....</b>	<b>241</b>
8.1	Groß- und Kleinschreibung.....	242
8.1.1	Variablen.....	244
8.1.2	Konstanten.....	245
8.1.3	Magische Konstanten.....	248
8.1.4	Funktionen und Methoden.....	249
8.1.5	Klassen.....	250
8.1.6	Dateien.....	251
8.2	Namenskonflikte.....	252
8.2.1	Reservierte Schlüsselwörter.....	253
8.2.2	Funktionen.....	256
8.2.3	Klassen, Interfaces und Exceptions.....	257
8.2.4	Konstanten.....	260
8.2.5	Magische Konstanten, Funktionen und Methoden.....	263
8.2.6	Variablen.....	263
8.2.7	Komponenten und Bibliotheken.....	265

8.3	Verarbeitung von Eingabedaten .....	268
8.3.1	Globale Variablen registrieren .....	268
8.3.2	Lange Eingabe-Arrays .....	271
8.3.3	Superglobale Variablen.....	271
8.3.4	Magic Quotes.....	272
8.3.5	Auf POST-Daten zugreifen.....	276
8.3.6	Skriptname: \$PHP_SELF .....	277
8.4	Fehlerbehandlung .....	278
8.4.1	Fehler unterdrücken mit @ .....	279
8.4.2	Die letzte Fehlermeldung speichern.....	280
8.4.3	Die Fehleranzeige konfigurieren.....	281
8.4.4	Neue und geänderte Fehlermeldungen .....	282
8.4.5	Eigene Fehler-Handler .....	285
8.4.6	Exceptions .....	287
8.5	Referenzen.....	287
8.5.1	Der PHP 4-Kompatibilitätsmodus .....	288
8.5.2	Eine Referenz erzeugen .....	289
8.5.3	Referenzen übergeben.....	290
8.5.4	Referenzen zurückgeben.....	292
8.5.5	Zur Laufzeit Referenzen übergeben.....	293
8.5.6	Objekte kopieren.....	294
8.6	Magische Konstanten .....	295
8.7	Geändertes Verhalten von PHP-Funktionen .....	297
8.7.1	array_merge() .....	297
8.7.2	ip2long() .....	299
8.7.3	strrpos().....	299
8.7.4	stripos() .....	300
8.7.5	strtotime() .....	300
8.8	Klassen .....	301
8.8.1	Statische Methoden und dynamische Aufrufe .....	301
8.8.2	Abstrakte private Methoden.....	302
8.8.3	Abstrakte statische Methoden .....	302
8.8.4	Geänderte Methodensignatur in abgeleiteten Klassen .....	303
8.9	Objekte.....	304
8.9.1	Konstruktor.....	304
8.9.2	Destruktoren .....	306
8.9.3	Klassenkonstanten neu definieren.....	307
8.9.4	instanceOf anstelle von is_a().....	307
8.9.5	Namenskonflikte mit \$this .....	308
8.9.6	\$this neu zuweisen .....	309
8.9.7	Objekte vergleichen .....	310
8.10	Dynamische Aufrufe .....	310
8.10.1	Systemaufrufe .....	311
8.10.2	Klassennamen .....	312
8.10.3	Die call_user_func()-Familie .....	313

8.10.4	Nachgeladener Code.....	315
8.10.5	eval .....	316
8.11	Kleine Einheiten.....	317
8.11.1	unset() und Zeichenketten.....	317
8.11.2	Fehler beim Senden der HTTP-Header.....	318
8.11.3	Datums- und Zeitfunktionen.....	320
8.11.4	Modulo-Division .....	320
8.11.5	Falsche Parameteranzahl in Funktionsaufrufen .....	321
8.11.6	Automatisches Umwandeln von Integer-Werten.....	323
8.11.7	Leere Objekte .....	324
8.11.8	\$this, Delegation und statische Aufrufe.....	324
8.11.9	Objekte ausgeben und die magische Methode __toString().....	326
8.12	PHP-Erweiterungen.....	326
8.12.1	mysql und mysqli .....	327
8.12.2	SPL.....	328
8.12.3	Tidy .....	328
8.12.4	Tokenizer.....	328
8.12.5	XML.....	328
	<b>Literatur .....</b>	<b>331</b>
	<b>Register .....</b>	<b>335</b>



## 2 Strategien

*Ob du eilst oder langsam gehst,  
der Weg bleibt immer der gleiche.*  
(Chinesisches Sprichwort)

In der Informationstechnologie gibt es keine endgültigen Lösungen. Ebenso wie die reale Welt ändert sich auch die Welt der Informationstechnologie stetig und schnell. Software wird erweitert, weil neue Features benötigt werden. Software wird optimiert, weil Ergebnisse schneller berechnet oder mehr Benutzer gleichzeitig bedient werden sollen. Es werden Fehler oder Sicherheitslücken entdeckt, also muss Software geändert werden.

Auch die Hardware ändert sich stetig. Nach Moores Gesetz [Droess 2005] verdoppelt sich ungefähr alle 18 Monate die Anzahl der Transistoren in einem Mikroprozessor<sup>1</sup>. Spätestens alle paar Jahre ändern sich Bauform und Schnittstellen von Hauptplatinen, Speichern oder Festplatten. Die Anwender erwarten selbstverständlich, dass Software nicht nur auf der neuesten verfügbaren Hardware, sondern auch auf älteren und bereits in die Jahre gekommenen Systemen läuft.

Für Anwendungsentwickler ist es nicht immer einfach, mit dieser Entwicklung Schritt zu halten. Dank der Arbeit von Betriebssystemherstellern und den Treiberprogrammierern bleibt ein Großteil der Hardware-Änderungen vor den Anwendungsentwicklern verborgen. Dafür müssen sie jedoch stärker auf das sich stetig ändernde Software-Umfeld Rücksicht nehmen. Je stärker eine Anwendung mit dem Betriebssystem verzahnt ist, desto schwieriger

---

<sup>1</sup> Moores Gesetz wird oft falsch zitiert. Nicht die Rechenleistung der Prozessoren verdoppelt sich alle 18 Monate, sondern die Anzahl der Transistoren. Natürlich bedeutet dies auch eine Steigerung der Rechenleistung, aber eben nicht unbedingt im gleichen Verhältnis.



ger ist es, bei gleichzeitiger Rückwärtskompatibilität zu älteren Versionen auch eine neue Betriebssystemversionen zu unterstützen.

Besonders schwer haben es die Systemadministratoren, die meist ein knappes Budget zur Verfügung haben, aber die Systeme immer wieder an ein sich schnell änderndes geschäftliches Umfeld anpassen müssen. Dabei müssen sie nach außen hin Kontinuität bewahren, da die Anwender ihre Gewohnheiten nur ungern ändern.

Die Migration von Software und Systemen spielt sich an der Schnittstelle zwischen Entwicklern und Administratoren ab. Um dabei erfolgreich zu sein, muss das zu migrierende System und dessen Umfeld ganzheitlich betrachtet werden. Auch die Anwender sollte man bei einer Migration niemals aus dem Blickfeld verlieren. Schließlich entscheiden sie, ob ein System benutzt wird oder nicht.

In diesem Kapitel beschäftigen wir uns mit verschiedenen grundlegenden Strategien zur Migration. Hierbei spielen Denkwänge der Systemverwaltung ebenso eine Rolle wie die Denkwänge der Softwareentwicklung. Aber auch die Anwender und das Management müssen sich trotz zu wenig Zeit und knapper Budgets darüber im Klaren sein, dass in der Informationstechnologie keine Lösung endgültig ist. Migrationen sind daher ein inhärenter Bestandteil des IT-Lebens.

Die Frage ist nicht, ob jemals wieder eine Migration notwendig sein wird, sondern wann die nächste Migration durchgeführt werden muss.

### 2.1 Never touch a running system

---

Jeder, der schon einmal ein System verwaltet hat, und sei es nur der Rechner am eigenen Arbeitsplatz, kennt das Problem: Bevor man sich in den Feierabend, ins Wochenende oder gar in den wohlverdienten Urlaub verabschiedet, soll nur noch schnell ein kleines Software-Update installiert, die Konfiguration des Rechners geändert oder ein neues Programm installiert werden.

Selbst wenn danach auf den ersten Blick die Welt noch in Ordnung scheint, ist damit nicht gesagt, dass nach dem nächsten Neustart des Systems noch alles wie erwartet funktioniert. Mitunter genügt eine Systemeinstellung, die nach einem Neustart nicht mehr wirksam ist oder erst durch einen Neustart wirksam wird, um Ihnen einige Stunden Stress und mühsame Fehlersuche zu bescheren.

Natürlich ist es reiner Aberglaube, aber manchmal könnte man den Eindruck gewinnen, der Rechner spürt es, wenn man es eilig hat oder kurz vor einem wichtigen Termin steht, und macht dann mit Vorliebe Ärger. Eine gern zitierte goldene Regel – wenn nicht *die* goldene Regel – der Informationstechnologie lautet daher: *Never touch a running system*.

Der Ursprung dieser Weisheit ist nicht bekannt. Ich könnte mir allerdings durchaus vorstellen, dass sie sich ursprünglich auf mechanische Geräte bezog und man einfach verhindern wollte, dass jemand einen Finger verliert. In der Informationstechnologie gibt es – vielleicht abgesehen von Gehäuselüftern – nur wenige direkte Risiken für Finger, aller-

dings riskiert man hier oft den sprichwörtlichen Kopf und Kragen, wenn man ein unternehmenskritisches, laufendes System „anfässt“ und daraus Produktionsausfall resultiert.

### 2.1.1 Systemumgebung

Mit *Running System* sind in erster Linie laufende Produktivsysteme gemeint. Sofern ein System nur zu normalen Bürozeiten verfügbar sein muss, ist es einfach, die notwendigen Wartungen abends oder am Wochenende durchzuführen, wenn das System nicht benötigt wird. Wichtig ist, dass man in diesem Fall einen genügend großen Zeitpuffer einplant, um das System bei eventuellen Problemen wieder auf Vordermann zu bringen, bevor es produktiv benötigt wird.

In der heutigen Internet-Zeit müssen viele Anwendungen 24 Stunden am Tag und sieben Tage pro Woche verfügbar sein. Zeiten der Nichtverfügbarkeit müssen je nach Einsatzgebiet des Systems lange vorher angekündigt werden, was eine langfristige Planung der Wartungstätigkeiten erfordert. Um die geplanten Ausfallzeiten möglichst kurz zu halten, ist man gut beraten, alle Maßnahmen zuerst auf möglichst identischen Testsystemen zu überprüfen, bevor man eine Änderung am Produktivsystem vornimmt.

Leider sind in der Realität die entsprechenden Testsysteme nicht immer vorhanden, und bei entsprechender Komplexität des Produktivsystems werden diese schnell zu einem Kostenfaktor. Man kann zwar heute durch Virtualisierung von Systemen gerade bei Testsystemen erhebliche Hardwarekosten sparen, aber je mehr Unterschiede es zwischen dem Testsystem und dem Produktivsystem gibt, desto wahrscheinlicher ist es, dass auf dem Produktivsystem neue oder andere Probleme auftauchen<sup>2</sup>.

Je älter Systeme werden, desto länger ist ihre Historie von installierter Software, aktualisierter Software, wieder entfernter Software und geänderter Konfiguration. Leider wird diese Historie in den seltensten Fällen wirklich sauber dokumentiert, sodass man meist nicht mehr mit Sicherheit sagen kann, mit welchen Installations- und Konfigurationsschritten man von der Grundinstallation zum aktuellen Systemzustand kommt. In Verbindung mit automatischen Updates von Programmen oder Systemkomponenten ist es natürlich besonders schwierig, den Überblick über den Versionsstand eines Systems zu behalten.

Je weniger ein solches System dokumentiert ist, desto besser ist man beraten, keine unnötigen, scheinbar harmlosen Änderungen daran vorzunehmen. Wenn nach einer solchen Änderung bestimmte Programme nicht mehr funktionieren und es nicht gelingt, das System wieder in einen Zustand zu bringen, in dem es wie erwartet funktioniert, steht man vor der Frage, wie man ein Ersatzsystem installieren und konfigurieren muss.

---

<sup>2</sup> In letzter Zeit gibt es den Trend, auch Produktivsysteme zu virtualisieren. Das macht es beispielsweise besonders einfach, eine wirklich identische Kopie eines Produktivsystems als Testsystem zu erstellen. Die virtuellen Maschinen haben eine einheitliche Hardware, die vom Wirtssystem weitgehend unabhängig ist. Dadurch sind die virtuellen Maschinen ohne Änderung auf unterschiedlichen physischen Systemen einsetzbar.

Sofern das ursprüngliche System – sozusagen als Vorlage – noch vorhanden und zugreifbar ist, ist es relativ einfach, im direkten Vergleich Alt gegen Neu ein identisches oder zumindest hinreichend ähnliches System zu installieren. Wenn Sie aber keine Ersatzhardware haben und deshalb das alte System tatsächlich neu installieren müssen, ist der Ausgang der Rettungsaktion ungewiss, zumal Sie sich jeglicher Rückfallposition berauben, sobald Sie die alte Installation überschreiben.

In jedem Fall müssen Sie ein Backup aller relevanten Daten und Einstellungen erstellen. Bei den heutigen Datenmengen kann es schon einige Stunden dauern, bis die relevanten Daten auf einen anderen Rechner übertragen sind. Am besten ist es natürlich, die originale Festplatte unversehrt aufzuheben und die neue Installation auf einer neuen Festplatte durchzuführen. Das funktioniert allerdings nur, wenn Sie direkten Zugriff auf den Rechner haben, und ist für Mietserver, die in Rechenzentren stehen, keine Option.

*Never touch a running system* wird oft als Entschuldigung dafür missbraucht, an einem System notwendige Updates und Aktualisierungen zu verschleppen. Es ist durchaus sinnvoll, an einem Produktivsystem nur Veränderungen vorzunehmen, die einen klaren Nutzen haben. Führen Sie dabei zuvor möglichst umfassende Tests auf einem anderen System durch, und sorgen Sie in jedem Fall dafür, dass Ihnen immer eine Rückfallposition bleibt, mit der Sie schnell und unkompliziert zu einem funktionierenden System zurückkehren können, falls tatsächlich etwas schiefgeht.

Sobald *Never touch a running system* zum Zwang wird, weil ein System nicht ausreichend dokumentiert ist, man nicht mehr weiß, was geschieht, wenn man am System Änderungen oder Aktualisierungen vornimmt, wird es sehr gefährlich, einfach weiter abzuwarten. Schon das nächste auftretende Problem kann durch eine ungünstige Verkettung von Abhängigkeiten Ihr gesamtes Systemkonzept wie ein Kartenhaus zusammenstürzen lassen, wie das folgende Beispiel zeigt.

Eines schönen Tages möchten Sie hinter dem Server sauber machen und beschließen, dabei gleich die Kabel zu ordnen. Sie fahren also den Server herunter, säubern die Ecke und schaffen dort Ordnung. Nach getaner Arbeit möchten Sie Ihren Server neu starten. Dieser verweigert allerdings den Dienst, da sich die Festplatte aus dem Leben verabschiedet hat<sup>3</sup>.

Sie müssen also eine neue Festplatte besorgen. Leider stellen Sie fest, dass Sie kurzfristig keine Festplatte mit der verwendeten Schnittstelle bekommen können. Leider unterstützt das Motherboard Ihres Servers die aktuellen Festplatten nicht mehr. Da Sie nicht darauf warten können, bis die Ersatzfestplatte geliefert wird, müssen Sie kurzfristig auf eine andere Server-Hardware ausweichen.

Leider lässt sich die vollständige Sicherung ihres alten Servers nicht so ohne Weiteres auf dem neuen Server installieren, da dort wegen der geänderten Hardware völlig andere Treiber nötig sind. Da Sie kein instabiles System riskieren wollen, entscheiden Sie sich dage-

---

<sup>3</sup> Tatsächlich fallen Festplatten eher selten im laufenden Betrieb aus, sondern eben beim Hochfahren eines Systems. Das ist ein Grund, warum Server möglichst selten heruntergefahren werden und dort die Stromsparmaßnahmen für die Festplatten abgeschaltet sein sollten.

gen, das vorliegende Backup einzuspielen, sondern beschließen, den neuen Server von Grund auf neu zu installieren.

Nach zwei bis drei vergeblichen Installationsversuchen müssen Sie einsehen, dass das etwas in die Tage gekommene Betriebssystem Ihres alten Servers auf der neuen Hardware nicht mehr installierbar ist. Selbst wenn Sie die Installation abschließen könnten, wäre es fraglich, ob das System jemals stabil läuft.

Sie sind also gezwungen, eine neue Betriebssystemversion einzusetzen. Diese lässt sich auf dem neuen Server problemlos installieren, sodass Sie neue Hoffnung schöpfen. Kurz darauf stellen Sie allerdings fest, dass nicht alle benötigten Softwarekomponenten, die Ihre Anwendung als Infrastruktur benötigt, unter dem neuen Betriebssystem laufen.

Leider können Sie nicht einfach die neueste Version bestimmter Softwarekomponenten mit einer älteren Version anderer Komponenten kombinieren, da sich zwischen den Versionen die internen Schnittstellen geändert haben. Sie müssen also einheitlich eine relativ aktuelle Version aller benötigten Softwarekomponenten installieren.

Nun kommt es, wie es kommen musste: Ihre Anwendung läuft in der neuen Umgebung nicht oder zumindest nicht fehlerfrei. Sie haben schon Stunden damit verbracht, ein funktionierendes System zu installieren, und müssen nun noch auf unbestimmte Zeit versuchen, Ihre Anwendung in der neuen Umgebung zum Laufen zu bekommen. Noch schlimmer ist es, wenn die Fehler nicht so offensichtlich sind, dass sie sofort entdeckt werden. In diesem Fall kann Ihnen eine sehr unruhige Zeit mit ungewissem Ausgang bevorstehen.

Eine solche Situation ist für alle Beteiligten wirklich unangenehm. Letztlich sind Sie gezwungen, alle Versäumnisse der Vergangenheit im Rahmen einer großen Feuerwehraction nachzuholen. Dies ist das Damoklesschwert, unter dem Sie stehen, wenn Sie zu lange nach der *Never touch a running system*-Philosophie leben.

### 2.1.2 Programmcode

Oftmals wird die Regel *Never touch a running system* auch als Rechtfertigung dafür verwendet, bestehenden Programmcode nicht zu ändern. Diese Sichtweise hat zunächst einmal durchaus ihre Berechtigung, denn jede auch noch so unbedeutend erscheinende Änderung am Quellcode kann dazu führen, dass ein Programm nicht mehr funktioniert beziehungsweise nicht mehr wie erwartet funktioniert.

Es ist leider viel zu einfach, bei einer kleinen Änderung am Quellcode einen Syntaxfehler in das Programm einzubauen. In übersetzten Sprachen deckt der Compiler solche Fehler schon frühzeitig auf, da der Quellcode nicht mehr übersetzt werden kann. In PHP allerdings gibt es keinen Compiler, daher bleiben diese Syntaxfehler zunächst unentdeckt und treten erst dann zutage, wenn die fehlerhafte Datei auch tatsächlich geladen und ausgeführt wird.

Eine gute integrierte Entwicklungsumgebung zeigt Ihnen direkt während des Editierens an, welche Syntaxfehler eine PHP-Datei enthält. Aber auch ohne IDE sollten Sie versuchen, wenigstens solche offensichtlichen Fehler möglichst frühzeitig zu finden. Führen Sie daher

# Register

## \$

`$_COOKIE` 74, 272  
`$_ENV` 74, 272  
`$_FILES` 272  
`$_GET` 74, 272  
`$_POST` 74, 272  
`$_REQUEST` 74, 84, 140, 272  
`$_SERVER` 74, 272, 277  
`$_SESSION` 81, 272  
`$argc` 268  
`$argv` 268  
`$HTTP_COOKIE_VARS` 271  
`$HTTP_ENV_VARS` 271  
`$HTTP_GET_VARS` 271  
`$HTTP_POST_FILES` 271  
`$HTTP_POST_VARS` 271  
`$HTTP_RAW_POST_DATA` 73, 276  
`$HTTP_SERVER_VARS` 271  
`$PATH` 44  
`$php_errormsg` 74, 280  
`$PHP_SELF` 277  
`$PHPRC` 70  
`$this` 308, 324

## %

`%PATH%` 44  
`%Windir%` 71

## .

`.htaccess` 65, 70, 85, 120, 188  
`.user.ini` 85

## \_

`__autoload()` 263  
`__call()` 263  
`__CLASS__` 248, 263  
`__clone()` 263, 295  
`__construct()` 263, 305  
`__destruct()` 263, 306  
`__FILE__` 248, 263, 277, 295  
`__FUNCTION__` 248  
`__LINE__` 248  
`__METHOD__` 248  
`__NAMESPACE__` 248  
`__sleep()` 263  
`__toString()` 326

## A

Access Control Lists siehe ACL  
ACL 39  
`addslashes()` 274  
Advanced Packaging Tool siehe APT  
AJAX 109, 166, 185, 217  
`allow_call_time_pass_reference` (php.ini)  
85, 147, 294  
`allow_url_fopen` (php.ini) 77, 83, 162  
`allow_url_include` (php.ini) 83, 315  
AllowOverride (Apache-Konfiguration)  
65, 70, 188  
Alternative PHP Cache siehe APC  
`always_populate_raw_post_data` (php.ini)  
73, 276  
AMD64 28

Apache 56  
apache (SAPI) 62  
Apache Ant 232  
Apache Portable Runtime siehe APR  
Apache.exe 119  
apache2handler (SAPI) 62  
APC 124, 127  
API-Dokumentation 232  
APR 57  
APT 32  
apt-get 166, 170, 176  
apxs 68  
arg\_separator.input (php.ini) 72  
arg\_separator.output (php.ini) 72  
Arial Unicode MS 47  
ARPANET 1  
array\_keys() 265  
array\_map() 274  
array\_merge() 297  
ASCII 45, 52, 106  
asp\_tags (php.ini) 72  
Assembler 27  
Assertion 215  
Asynchronous JavaScript and XML  
  siehe AJAX  
at 55, 162  
Ausgabepufferung  
  siehe Output Buffering  
Ausnahme siehe Exception  
auto\_append\_file (php.ini) 82, 191  
auto\_detect\_line\_endings (php.ini) 73  
auto\_prepend\_file (php.ini) 82, 191

## B

Backslash 40  
Backup 53, 114  
Backwards Compatibility 241  
bad interpreter: No such file or directory  
  (Fehlermeldung) 37  
Bakken, Stig 192  
Befehlssatz 27  
Befehlssatz siehe Instruction Set  
Berners-Lee, Tim 2

Big Endian 30  
Bison 68  
Bit 28  
BLOB 51, 112  
Bochs 208  
BOM 102  
BOOLEAN 51  
Breakpoint 228  
Browserkrieg 182  
build.xml 234  
Byte 28  
Byte Order Mark siehe BOM  
Byte-Reihenfolge siehe Endian-ness

## C

call\_user\_func() 313  
call\_user\_func\_array() 313  
call\_user\_method() 313  
call\_user\_method\_array() 313  
Call-time pass-by-reference has been  
  deprecated (Fehlermeldung) 294  
Cannot modify header information  
  (Fehlermeldung) 319  
Cannot re-assign \$this  
  (Fehlermeldung) 307, 309  
Cascading Stylesheets siehe CSS 166  
case-insensitive 41  
case-sensitive 41  
case-sensitive siehe Groß- und  
  Kleinschreibung 242  
catch() 287  
Catchable fatal error  
  (Fehlermeldung) 326  
CGI 3, 59, 64, 65, 80, 85, 86, 93, 94, 122  
cgi (SAPI) 62  
CGI-Umgebungsvariablen 66  
CGI-Wrapper 59, 87  
chgrp 39  
chmod 39  
chown 39  
chroot 59  
chroot-Jail 59, 87, 99  
CLI 71

- cli (SAPI) 62, 93
- CLOB 112
- CLOB 51
- clone 163, 294
- Code Coverage 231, 237
- Codeabdeckung siehe Code Coverage
- Codepoint 45
- Coding Guidelines 154
- Coding Style 199
- Coggeshall, John 172
- Collation 52
- COM 100
- Comprehensive Perl Archive Network  
  siehe CPAN
- constant() 262
- Content-Type (HTTP-Header) 75, 106, 110
- Corba 100
- Core siehe Rechenkern
- CPAN 5, 192
- cpuinfo 118
- cron 55, 162
- CSS 166, 174, 186
- current() 139
- CVS 160
  
- D**
- daemon 58
- DATE 51
- date() 320
- date.timezone (php.ini) 83, 320
- date\_default\_time\_zone() 320
- Datei
  - Name 41
- Dateien
  - durchsuchen 163
  - herunterladen 161
  - vergleichen 164
- Dateiname 41
- Dateirechte 38
- Dateiupload 80
- Datei-URL siehe File-URL
- Datenbank
  - binäres Backup 54
  - Export 54
  - Hot-Backup 54
  - SQL-Dump 54
- Datensicherung siehe Backup
- Datentyp 51
- DCOM 100
- Debian 32
- Debugging
  - JavaScript 185
- Declaration should be compatible with  
  (Fehlermeldung) 304
- dedizierter Server 63
- default\_charset (php.ini) 75
- default\_mimetype (php.ini) 75
- default\_socket\_timeout (php.ini) 78
- define() 246, 253, 261
- Destruktor
  - simulieren 306
- Destruktor siehe \_\_destruct()
- DHCP 210
- diff 119, 164
- disable\_classes (php.ini) 78
- disable\_functions (php.ini) 78
- Diskette 47
- display\_errors (php.ini)
  - 74, 133, 280, 281, 285
- Distribution 32
- Distribution siehe Linux-Distribution
- dl() 86, 89
- DNS 150
- DOCTYPE 169
- Document Type Definition siehe DTD
- DOM 167, 184, 213, 329
- DOM-Baum 166
- DOMDocument 178
  - loadXML() 178
- dos2unix 37
- DSO 60
- DTD 167, 169, 176
- Dynamic Shared Objects siehe DSO
- dynamisch ladbares Modul siehe DSO

## E

E\_ALL 284  
 E\_DEPRECATED 284  
 E\_ERROR 286  
 E\_NOTICE 246, 286  
 E\_STRICT 84, 85, 133, 145, 190, 282, 286,  
 303, 305, 315, 325  
 E\_WARNING 286  
 eAccelerator 124  
 Eclipse PDT 228  
 Eingabedaten 268  
 enable\_dl (php.ini) 86  
 Endian-ness 30  
 error\_log (php.ini) 144  
 error\_reporting (php.ini) 74, 133, 190, 281,  
 284, 285  
 error\_reporting() 294  
 eval() 145, 316  
 Exception 257, 287  
 exec() 97, 311  
 extension (php.ini) 92, 173  
 extension\_dir (php.ini) 92, 173  
 extension\_loaded() 89  
 ezComponents 95  
 ezPublish 95

## F

fail() 137  
 FastCGI 64, 66, 69, 85, 86, 93, 94, 122  
 Fatal error: Abstract function cannot be  
 declared private (Fehlermeldung) 302  
 Fatal error: Cannot redeclare class  
 (Fehlermeldung) 251, 259, 315  
 Fatal Error: Cannot redeclare class  
 (Fehlermeldung) 266  
 Fatal error: Cannot redefine class constant  
 (Fehlermeldung) 307  
 Fatal error: Cannot unset string offsets  
 (Fehlermeldungen) 318  
 Fatal error: Class not found  
 (Fehlermeldung) 316

Fatal error: Method Test::\_\_toString() must  
 not throw an exception  
 (Fehlermeldung) 326  
 Fatal error: Nesting level too deep - recursive  
 dependency? (Fehlermeldung) 310  
 Fatal error: Unknown: Failed opening  
 required 191  
 file() 82  
 file\_exists() 280  
 file\_get\_contents() 77, 82, 83, 162, 252,  
 277, 280  
 file\_uploads (php.ini) 78  
 File-URL 41  
 filter.default (php.ini) 84  
 filter.default\_flags (php.ini) 84  
 Firebug 185  
 Firefox 109, 182  
 Firewall 58, 61, 113, 131  
 Fixture 213  
 Flex 68  
 Fließkommazahl 29, 51, 75  
 FLOAT 51  
 Font siehe Schriftart  
 fopen() 77, 82, 244, 252  
 Forward Slash 40  
 fsockopen() 78  
 FTP 64

## G

Ganzzahl 29  
 GCIwrap 59  
 GD 34  
 gespeicherte Prozedur  
 siehe Stored Procedure  
 get\_cfg\_var() 188  
 get\_class() 257  
 get\_current\_user() 120  
 get\_declared\_classes() 258  
 get\_declared\_interfaces() 258  
 get\_defined\_constants() 261  
 get\_defined\_functions() 257  
 get\_defined\_vars() 265  
 get\_extension\_funcs() 257



- get\_loaded\_extensions() 89
- get\_magic\_quotes\_gpc() 274
- get\_magic\_quotes\_runtime() 275
- getmygid() 120
- getmyuid() 120
- global 271
- go-pear.phar 162, 192, 193
- grep 163
- Groß- und Kleinschreibung 242
  - CamelCase 245
  - Dateinamen 251
  - Funktionen 249
  - Klassen 250
  - Konstanten 245
  - Methoden 249
  - URL 242
  - Variablen 244
- Gutmans, Andi 4
- H**
- header() 106, 319
- Hex-Editor 103
- htdocs 41
- HTML 2
- HTML Tidy 170
  - CleanRepair() 174
  - diagnose() 174
  - parseString() 174
  - PHP-Erweiterung 172
- HTML-Entities 102
- HTML-Standard 166
- HTTP
  - Header 318
- HTTP-Authentication 224
- HTTP-Authentifizierung 64, 66
- httpd 119
- httpd.conf 61, 64, 66, 67, 70, 119, 187
- Hypertext 2
- Hypertext Markup Language siehe HTML
- I**
- i386 26
- IA32 26, 208
- IBM-kompatibel 26
- iconv 109
- iconv() 106
- iconv\_substr() 109
- IDE 161
- ignore\_user\_abort (php.ini) 78
- ignore\_user\_abort() 78
- IIS 56
- IMAP 34
- import\_request\_variables() 270
- include 77, 82, 83, 215, 252, 266, 315
- Include (Apache-Konfiguration) 119
- include\_path (php.ini) 44, 82, 121
- ini\_get() 188
- ini\_get\_all() 120, 188, 189
- ini\_set() 70, 84, 120, 121, 275, 280, 282
- instanceOf 282, 308
- INT 51
- Integrationstest 148
- Integrierte Entwicklungsumgebung siehe IDE
- Interface 257
- Internet 1
- Internet Explorer 109, 182
- Internet Information Services siehe IIS
- Internet Protocol siehe IP
- IP 31, 210
- ip2long() 299
- is\_a() 282, 307
- is\_file() 257
- is\_readable() 280
- isapi (SAPI) 62
- ISO 8859 45, 102
- ISO-Image 211
- Iterator 258, 328
- J**
- JAR 221
- Java 181, 220
- JavaScript 166, 213
- Jigsaw 175
- jsl 179
- JSLint 180

## K

Kern siehe Rechenkern  
 Kollation siehe Collation  
 Kommandozeile  
     Aufruf 98  
     Parameter 99  
 Konstanten  
     eingebaute 247  
 Konstruktor siehe \_\_construct()  
 Kontrollfaden siehe Thread

## L

Laufzeitfehler 143  
 Lerdorf, Rasmus 3, 49  
 libmysql.dll 327  
 libxml.dll 177  
 libxml2 176  
 LIMIT 49  
 Lint 143, 145, 176, 190, 233  
     JavaScript 179, 180  
     PHP 190  
 Little Endian 30  
 LoadModule (Apache-Konfiguration) 63  
 Locking Strategy 48  
 log\_errors (php.ini) 144  
 Logdatei 114  
 LSB 33  
 LSB siehe Linux Standard Base

## M

Mac OS X 32, 179  
 MAC-Adresse 132  
 magic\_quotes (php.ini) 272  
 magic\_quotes\_gpc (php.ini) 86, 147, 274  
 magic\_quotes\_runtime (php.ini) 86, 147, 275  
 magic\_quotes\_sybase (php.ini) 86, 147, 275  
 Magische Konstanten 248  
 mail() 76, 207  
 mail.force\_extra\_parameters (php.ini) 76  
 Maschinensprache 35  
 max\_execution\_time (php.ini) 79  
 max\_input\_nesting\_level (php.ini) 84

max\_input\_time (php.ini) 79  
 mbstring 109  
 Mehrbyte-Zeichensatz 52, 109  
 Meld 166  
 memory\_limit (php.ini) 79, 80  
 Mesh 2  
 Meta-Tag 110  
 MIME-Typ 75  
 Mock-Objekt 216  
 mod\_action 66  
 mod\_rewrite 188  
 mount 40  
 move\_uploaded\_file() 276  
 MPM siehe Multi-Processing-Modul  
 Multi-Processing-Modul 34, 57  
 Multitasking 33  
 Multithreading 33, 58, 68  
 Multithreading 57  
 MX-Lookup 76  
 my.cnf 118  
 my.ini 118  
 MySQL 54, 124, 327  
     Client-Bibliothek 327  
 mysql\_real\_escape\_string() 276  
 mysqld 118  
 mysqld-nt 118  
 mysqldump 55  
 mysqlnd 327

## N

Namenspräfix 255  
 Namensräume siehe Namespaces  
 Namespaces 156, 253  
 Netscape 182  
 new 289, 302, 312  
 Non-static method should not be called  
     statically (Fehlermeldung) 325  
 Normalisierung 48  
 Notfallplan 115  
 Notice: Constant already defined  
     (Fehlermeldung) 246, 261

- Notice: Only variable references should be returned by reference (Fehlermeldung) 293
- Notice: Use of undefined constant (Fehlermeldung) 247
- NTFS 112
- ## O
- ob\_clean() 282
- ob\_end\_clean() 174
- ob\_start() 173, 282
- Opcache-Cache 4
- open\_basedir (php.ini) 80
- OpenType 46
- Options (Apache-Konfiguration) 188
- Output Buffering 317, 319
- Overflow 29
- ## P
- Parallels 208
- Parse error: syntax error (Fehlermeldung) 253
- Parse error: syntax error, unexpected ... (Fehlermeldung) 254, 279
- Parsebaum 253
- passthru() 311
- PATH\_SEPARATOR (Konstante) 44, 82
- pci 206
- pcrc.backtrack\_limit (php.ini) 84
- pcrc.recursion\_limit (php.ini) 84
- PDF 125
- PDO 92
- pear
- channel-update 194
  - help 194
  - install 196
  - list 193
  - upgrade 194
- PEAR 5, 89, 95, 162, 192
- pear version 192
- PEAR\_Mail 77
- PECL 68, 88, 89
- pecl install 92, 225
- Pederick, Chris 182
- Personal Homepage Tools 3
- phing 143, 233
- php
- d 190
  - i 188
  - l 190
  - m 225
- PHP
- 4.0 4
  - 4.1 277
  - 4.4 127
  - 5.0 5, 303
  - 5.2 77, 83, 84
  - 5.2 144
  - 5.3 8, 156, 266
  - 6.0 7, 8, 46, 109, 147, 156
- PHP Extension and Application Repository
- siehe PEAR
- PHP Extension Community Library
- siehe PECL
- PHP Fatal error: Cannot redeclare (Fehlermeldung) 249
- PHP Hypertext Preprocessor 4
- php -m 120
- php.exe 193
- php.ini 70, 87, 88, 187, 191
- php.ini-recommended 120, 147
- PHP/FI 3
- php\_admin\_flag (httpd.conf) 64
- php\_admin\_value (httpd.conf) 64
- php\_beautifier 196
- PHP\_Beautifier 195, 199
- php\_beautifier.bat 197
- php\_check\_syntax() 190
- PHP\_CodeSniffer 199
- PHP\_CodeSniffer\_File
- getTokens() 203
- PHP\_CodeSniffer\_Sniff (Interface) 203
- register() 203
- PHP\_Compat 194
- loadFunction() 195
  - loadVersion() 195

- PHP\_CompatCheck 205
  - PHP\_CompatInfo 89, 121, 122
  - PHP\_EOL (Konstante) 36, 247
  - php\_flag (.htaccess) 65
  - PHP\_INI\_USER 189
  - PHP\_INT\_MAX (Konstante) 35
  - PHP\_INT\_SIZE (Konstante) 35
  - PHP\_OS (Konstante) 296
  - PHP\_SAPI (Konstante) 63, 70, 261
  - php\_sapi\_name() 63, 70
  - php\_value (.htaccess) 64
  - PHP\_VERSION (Konstante) 62
  - php-<sapi>.ini 70
  - phpcs 200
  - PHP-Erweiterung 122
  - PHP-Erweiterung siehe PHP-Extension
  - PHP-Extension 87
  - phpinfo() 34, 62, 63, 88, 119, 188
  - PHPIniDir (Apache-Konfiguration) 70
  - phpize 93
  - phpMyAdmin 55, 118
  - PHPUnit 142, 144, 213, 235
  - PHP-Version 62
  - phpversion() 62
  - physische Grenze 111
  - ping 98
  - Plex86 208
  - popen() 311
  - Port
    - 443 61
    - 80 61
  - POSIX 120
  - posix\_getuid() 120
  - post\_max\_size (php.ini) 80
  - Postfix 76
  - PowerPC 26, 208
  - precision (php.ini) 75
  - Prefork MPM 57
  - Prefork-MPM 34
  - preg\_last\_error() 84
  - proc\_open() 311
  - Programmierrichtlinien
    - siehe Coding Guidelines
  - Programmierrichtlinien siehe Coding Style
  - Proxy 131
  - Prozess 33
- Q**
- QEMU 208
  - qmail 76
  - query() 139
- R**
- readfile() 82
  - Rechenkern 27, 33
  - Rechnerarchitektur 25
  - Redefining already defined constructor
    - (Fehlermeldung) 306
  - Refactoring 124, 156
  - Referenzen 287
    - zur Laufzeit übergeben 293
  - register\_argc\_argv (php.ini) 73
  - register\_globals (php.ini) 73, 147, 192, 268, 277
  - register\_long\_arrays 271
  - register\_long\_arrays (php.ini) 74, 147
  - Rekursion 226
  - require 77, 82, 83, 191, 215, 252, 266, 315
  - require\_once 195
  - reservierte Wörter 42
  - Restore 53
  - Rethans, Derick 225
  - Rhino 180
  - Root Directory 41
  - RPM 33
  - Rückwärtskompatibilität
    - siehe Backwards Compatibility
- S**
- Safe Mode 86
  - safe\_mode (php.ini) 86, 147
  - Sandbox 101
  - SAPI 62, 68, 70
  - Schlüsselwort 253
  - Schriftart 46
  - schtasks 55, 162

- Scope 263
- sed 164
- Selenium 146, 148, 216
- Selenium IDE 142, 144
- Semantic Web 2
- Sendmail 76
- sendmail\_from (php.ini) 76
- sendmail\_path (php.ini) 76
- SEQUEL 48
- serialize\_precision (php.ini) 76
- Server API siehe SAPI
- serviceorientierte Architektur siehe SOA
- Session 81
- session.auto\_start (php.ini) 81
- session.bug\_compat\_42 (php.ini) 81
- session.bug\_compat\_warn (php.ini) 81
- session.cookie\_domain (php.ini) 82
- session.cookie\_httponly (php.ini) 85
- session\_register() 81
- session\_start() 81, 319
- SET NAMES 53
- set\_error\_handler() 133, 285
- set\_magic\_quotes\_runtime() 275
- set\_time\_limit() 79
- setcookie() 319
- setrawcookie() 319
- setUp() 136, 142
- SFTP 64
- SGML 167
- Shared Hosting 63, 65, 89, 94
- Shebang 36, 65
- Shell 99
- shell\_exec() 311
- short\_open\_tags (php.ini) 147
- Shutdown-Handler 306
- SID 132
- SimpleTest 144
- single-threaded 58
- Single-Threading 68
- Slash siehe Forward Slash
- SMTP 77
- Snapshot 131, 148, 209, 211
- SOA 100
- SOAP 100, 121, 125, 145
- Speicherwort 28
- Sperrstrategie siehe Locking Strategy
- SPL 258
- SPLFileObject
  - getFilename() 328
  - getPathname() 328
- SPLFileObject 328
- SQL 48
- SQLite 138
- sqlite\_escape\_string() 276
- SQLiteDatabase 139
- Standard PHP Library siehe SPL
- statisches Modul 60
- Stored Procedure 50
- str\_replace() 296
- Strict Standards: Assigning the return value of new by reference is deprecated (Fehlermeldung) 289
- Strict standards: Declaration of should be compatible (Fehlermeldung) 304
- Strict standards: Function call\_user\_method() is deprecated (Fehlermeldung) 314
- Strict standards: is\_a(): Deprecated. Please use the instanceof operator (Fehlermeldung) 308
- Strict standards: It is not safe to rely on the system's timezone (Fehlermeldung) 320
- Strict standards: Non-static method should not be called statically (Fehlermeldung) 301
- Strict standards: Static function should not be abstract (Fehlermeldung) 302
- stripslashes() 274, 276
- stristr() 244
- stripos() 300
- strrpos() 299
- strstr() 244
- strtolower() 248
- strtotime() 300
- substr() 108, 245, 300, 323
- Subversion 160
- Suchpfad 44

suExec 59, 66  
Sun SPARC 26  
Superglobale Variablen 271  
SuSE 32  
Syntax-Highlighting 255  
Syntaxprüfung  
    JavaScript 178, 179, 180  
system() 97, 311  
systeminfo 118

### T

T\_ML\_COMMENT 328  
T\_PAAMAYIM\_NEKUDOTAYIM 255  
Task siehe Prozess  
tearDown() 137, 142  
temporäres Verzeichnis 43  
Testinventar siehe Fixture  
Testmodus 100, 101  
this siehe \$this  
Thread 33  
Thread-Modell 68, 119  
Thread-Modell siehe Thread  
Thread-Sicherheit siehe Thread-Safety  
throw 287  
Tidy 174, 328  
    CleanRepair() 174  
    diagnose() 174  
Tidy siehe HTML Tidy  
TIME 51  
TIMESTAMP 51  
Token 253  
token\_get\_all() 164  
Tokenizer 328  
Tracelog 125  
Trace-Log 227  
track\_errors (php.ini) 74, 280  
trigger\_error() 285  
TrueType 46  
try{} 287  
Turck MMCache 124, 127

### U

Überlauf siehe Overflow

Ubuntu 32  
Unicode 7, 45, 52  
Uniform Resource Locator siehe URL  
Unit-Tests 213  
Unix 32  
unix2dos 37  
unset() 317  
upload\_max\_filesize (php.ini) 80  
upload\_tmp\_dir (php.ini) 83  
URL 242  
USB-Stick 47  
user\_ini.filename (php.ini) 85  
UTF-16 46, 102  
UTF-32 46  
UTF-8 46, 101

### V

validieren  
    (X)HTML 168  
    XML 176  
var\_dump() 226  
VARCHAR 51  
variables\_order (php.ini) 74  
version\_compare() 62  
Versionsverwaltung 160  
Verzeichnis (temporäres)  
    siehe temporäres Verzeichnis  
Verzeichnispfad 41  
Verzeichnistrennzeichen 40  
VirtualBox 208  
VirtualPC 208  
Virtuelle Maschine 207  
VMware 131, 208  
    Workstation 211  
VMware Converter 131  
von Neumann, John 25

### W

W3C 168, 170, 182  
Wagenrücklauf siehe Zeilenendezeichen  
Warning: array\_merge(): Argument #2 is not  
    an array (Fehlermeldung) 297

Warning: Division by zero  
 (Fehlermeldung) 321  
 Warning: failed to open stream  
 (Fehlermeldung) 252  
 Warning: failed to open stream: No such file  
 or directory (Fehlermeldung) 280  
 Warning: Wrong parameter count  
 (Fehlermeldung) 322  
 Webdeveloper Toolbar 182  
 web-developer.xpi 182  
 well-formed 176  
 Wertebereich 29  
 Wertebereich siehe Scope  
 wget 162  
 Wiederherstellung siehe Restore  
 Windows 32  
 Winmerge 165  
 gewinnt MPM 57  
 wohlgeformt siehe well-formed  
 wordwrap() 257  
 Worker MPM 34, 57  
 World Wide Web Consortium siehe W3C  
 Wortlänge 28, 35  
 Wurzelverzeichnis siehe Root Directory  
 WWW siehe World Wide Web  
 wwwroot 41  
 wwwrun 58

**X**

x86 208, 209  
 xdebug 122, 125, 133, 225  
 xdebug.auto\_trace (php.ini) 227  
 xdebug.collect\_params (php.ini) 227  
 xdebug.collect\_return (php.ini) 227  
 xdebug.remote\_enable (php.ini) 228  
 xdebug.remote\_handler (php.ini) 228

xdebug.remote\_host (php.ini) 228  
 xdebug.remote\_port (php.ini) 228  
 xdebug.show\_mem\_delta (php.ini) 227  
 xdebug.so 225  
 xdebug.trace\_output\_dir (php.ini) 227  
 xdebug\_get\_function\_stack() 286  
 XEN 208  
 XHTML 166, 167  
 XML 72, 123, 125, 176, 184, 234, 328  
 XMLHTTP 186  
 xmllint 176  
 xmllint.exe 177  
 XMLReader 123, 207  
 XMLRPC 100  
 XMLWriter 123  
 XPath 184  
 XPather 184  
 xpi 183, 184, 185  
 XSL 329  
 xvfb 221

**Z**

ze1\_compatibility\_mode (php.ini) 147  
 Zeichensatz 44, 51  
 Zeilenendezeichen 35, 38  
   Macintosh 73  
 Zeilenvorschub siehe Zeilenendezeichen  
 Zend 4, 95  
 Zend Framework 95  
 zend.ze1\_compatibility\_mode (php.ini) 87,  
 288  
 zend\_extension\_ts (php.ini) 225  
 Zugriffskontrolllisten siehe ACL  
 Zugriffsrechte 38, 39  
 Zugriffsrechte siehe Dateirechte  
 Zusicherung siehe Assertion