



Oliver Kreipl, Dorian Karnbaum, Marc Remolt, Heiko Marr † © webmasters akademie Nürnberg GmbH, Nürnberg, Germany

Webserver-Administration mit Apache, PHP und MySQL

Ein Webmasters Press Lernbuch

Version 1.1.0 vom 20.6.2017

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm.

www.webmasters-europe.org

Inhaltsverzeichnis

Vorwort	11
1 Allgemeine Einführung	12
1.1 Benötigte Vorkenntnisse.	12
1.2 Technische Voraussetzungen	12
2 Installation von Apache, PHP und MySQL	13
2.1 LAMP	13
2.1.1 Linux	13
2.1.2 Apache	13
2.1.3 PHP	14
2.1.4 MySQL	14
2.2 Installation der LAMP-Komponenten	15
2.2.1 Backports von dotdeb.org	15
2.2.2 Installation	17
2.2.3 Testen der Installation	17
2.3 Zusammenfassung	18
2.4 Testen Sie Ihr Wissen	18
3 Deployment von Webseiten	19
3.1 Was ist Deployment?	19
3.2 Deployment statischer Webseiten	19
3.2.1 Document-Root	19
3.3 Deployment dynamischer Webseiten	20
3.3.1 PHP-Skripte	20
3.3.2 Datenbank	21
3.3.3 Konfiguration	22
3.4 CMS Made Simple	22
3.4.1 Herunterladen und installieren der Anwendungs-Dateien	23
3.4.2 Anlegen der Datenbank	23
3.4.3 Konfiguration über das Webinterface	23
3.5 Zusammenfassung	27
3.6 Testen Sie Ihr Wissen	28
4 PHP-Administration	29
4.1 PHP für Kommandozeile und Webserver	29
4.1.1 Installation von php5-cli	29
4.1.2 PHP-Skripte als Shellkommandos	29
4.2 Grundkonfiguration	30
4.2.1 Konfiguration für Webserver und Kommandozeile	30
4.2.2 Allgemeine Einstellungen	31
4.2.3 Ressourcenverwaltung	33
4.3 Konfiguration von PHP-Erweiterungen	34

4.3.1	Konzept	34
4.3.2	Session	34
4.3.3	Installation von PHP-Erweiterungen über apt	37
4.4	Zusammenfassung	39
4.5	Testen Sie Ihr Wissen	40
5	Einführung in MySQL	41
5.1	Verzeichnisstruktur von MySQL auf Debian	41
5.1.1	/etc/mysql	41
5.1.2	/var/lib/mysql	41
5.2	MySQL-Kommandozeilenclient	42
5.2.1	Verbindung zum Datenbankserver aufbauen	42
5.2.2	Befehle an MySQL senden	43
5.3	SQL für Administratoren	44
5.3.1	Konzept	44
5.3.2	Datenbanken erzeugen und löschen	44
5.3.3	Auslesen von Informationen mit SHOW	45
5.3.4	Datensätze auslesen	49
5.4	Zusammenfassung	50
5.5	Testen Sie Ihr Wissen	50
6	MySQL: Benutzer und Rechte	51
6.1	Benutzer und Rechte in MySQL	51
6.1.1	Die Datenbank mysql	51
6.2	Benutzerverwaltung	52
6.2.1	Passwort ändern	52
6.2.2	Benutzer löschen	54
6.2.3	Benutzer anlegen	55
6.3	Rechteverwaltung	56
6.3.1	Rechte ansehen	56
6.3.2	Rechte gewähren	58
6.3.3	Rechte nehmen	61
6.4	Zusammenfassung	62
6.5	Testen Sie Ihr Wissen	62
7	MySQL: Backup und Recovery	63
7.1	MySQL-Datenbanken reparieren	63
7.1.1	Beschädigte Tabellen	63
7.1.2	mysqlcheck	63
7.1.3	mysqlrepair	64
7.2	Backup von MySQL-Datenbanken	65
7.2.1	Notwendigkeit	65
7.2.2	mysqldump	65
7.2.3	Backups mit mysqldump automatisieren	66
7.2.4	Replikation	67
7.3	Einspielen von Sicherungen	68
7.3.1	Gründe für eine komplette Wiederherstellung	68
7.3.2	Einspielen von Sicherungen	68

7.3.3	Wiederherstellung einer Datenbank	68
7.3.4	Wiederherstellung des kompletten MySQL-Servers	69
7.4	Zusammenfassung	72
7.5	Testen Sie Ihr Wissen	73
8	Das HTTP-Protokoll	74
8.1	Aufbau des HTTP-Protokolls	74
8.1.1	Beispiel für eine Anfrage über HTTP	74
8.1.2	Versionen des HTTP-Protokolls	75
8.2	HTTP-Request	76
8.2.1	Request-Methoden	77
8.2.2	Client-Request-Header	78
8.3	HTTP-Response	79
8.3.1	Server-Response-Header	80
8.4	Zusammenfassung	81
8.5	Testen Sie Ihr Wissen	81
9	Konfiguration von Apache	82
9.1	Struktur der Apache-Konfiguration	82
9.1.1	apache2.conf	82
9.1.2	ports.conf	82
9.1.3	envvars	82
9.1.4	conf-available, conf-enabled	83
9.1.5	mods-available, mods-enabled	83
9.1.6	sites-available, sites-enabled	83
9.2	Apache-Dokumentation	83
9.3	Grundkonfiguration	84
9.4	Apache-Module	84
9.4.1	Konzept	84
9.4.2	Aktivierung eines Moduls	84
9.4.3	Deaktivierung eines Moduls	86
9.5	Zusammenfassung	87
9.6	Testen Sie Ihr Wissen	87
10	Virtuelle Hosts	88
10.1	Virtuelle Hosts	88
10.1.1	Konzept	88
10.1.2	Typen von virtuellen Hosts	88
10.2	Konfiguration namensbasierter virtueller Hosts	90
10.2.1	Voraussetzungen	90
10.2.2	NameVirtualHost-Direktive	92
10.2.3	VirtualHost-Container	93
10.2.4	Aktivieren und Deaktivieren von virtuellen Hosts	96
10.2.5	Optionale Direktiven	97
10.2.6	Options	100
10.3	Zusammenfassung	102
10.4	Testen Sie Ihr Wissen	102

11	Konfigurations-Bereiche	103
11.1	Geltungsbereich von Apache-Anweisungen	103
11.1.1	Globale Konfiguration	103
11.1.2	VirtualHost-Konfiguration	104
11.1.3	Konfiguration auf Verzeichnisebene	104
11.1.4	Konfiguration für einzelne Dateien	104
11.1.5	Anwendungsbereiche einzelner Anweisungen	104
11.2	Directory-Container	105
11.3	Location-Container	106
11.4	Files-Container	106
11.5	.htaccess-Dateien	107
11.5.1	Konzept	107
11.5.2	AllowOverride	108
11.5.3	.htaccess-Dateien editieren	109
11.6	Zusammenfassung	110
11.7	Testen Sie Ihr Wissen	110
12	Zugriffskontrolle	111
12.1	Anforderungen	111
12.2	Hostbasierte Zugriffsbeschränkung	111
12.2.1	Konzept	111
12.2.2	Allow und Deny	112
12.2.3	Order	113
12.3	Require	114
12.3.1	RequireAny-Container	116
12.3.2	RequireAll-Container	116
12.4	Beschränkung durch Authentifizierung	117
12.4.1	Konzept	117
12.4.2	Konfiguration	117
12.4.3	Erstellen der Passwort-Datei	119
12.5	Zusammenfassung	120
12.6	Testen Sie Ihr Wissen	121
13	URL-Manipulation	122
13.1	Zusammenhang zwischen URL und Dateipfad	122
13.2	Alias	123
13.3	Redirect	124
13.4	Reguläre Ausdrücke	126
13.4.1	Einführung	126
13.5	URL-Manipulation mit mod_rewrite	126
13.5.1	Konzept und Installation	126
13.5.2	RewriteEngine	127
13.5.3	RewriteRule	127
13.5.4	RewriteCond	129
13.6	Zusammenfassung	132
13.7	Testen Sie Ihr Wissen	132

14	Virtuelle Hosts mit SSL/TLS	133
14.1	HTTPS	133
14.2	Erstellen von SSL-Zertifikaten	133
14.3	Konfiguration von Apache	136
14.3.1	HTTPS-Port	136
14.3.2	mod_ssl	136
14.3.3	VHost-Konfiguration	136
14.3.4	SSL und IP-Adressen	139
14.4	Zusammenfassung	140
14.5	Testen Sie Ihr Wissen	140
15	Log-Analyse	141
15.1	Anforderungen	141
15.2	Informationen aus Logdateien	141
15.3	Webalizer	141
15.4	Piwik	142
15.5	AWStats	142
15.6	Konfiguration von AWStats	143
15.6.1	Installation des Programms	143
15.6.2	Ausführen von CGI-Skripten	143
15.6.3	Konfiguration von AWStats	144
15.6.4	Analyse der Logdatei	145
15.6.5	Regelmäßige Aktualisierung der Analysedaten	146
15.7	Schützen der Statistikseite	147
15.8	Zusammenfassung	147
15.9	Testen Sie Ihr Wissen	147
16	Sicherheit	148
16.1	Sicherheit in Apache	148
16.1.1	Listen	148
16.1.2	Dateisystem	149
16.1.3	ServerTokens	151
16.1.4	Includes	152
16.1.5	CGI-Skripte	153
16.2	Zusammenfassung	154
16.3	Testen Sie Ihr Wissen	154
	Lösungen der Übungsaufgaben	155
	Lösungen der Wissensfragen	170
	Index	180

Konfigurations-Bereiche

11

In dieser Lektion lernen Sie

- ▶ dass Sie Apache-Anweisungen für verschiedene Bereiche in der Konfiguration setzen können.
- ▶ wie Sie globale Einstellungen für bestimmte Bereiche überschreiben.
- ▶ wie Sie es Kunden erlauben können, bestimmte Einstellungen selbst vorzunehmen.

11.1 Geltungsbereich von Apache-Anweisungen

Bisher haben Sie sich mit der Konfiguration von Apache vertraut gemacht. Sie haben Module aktiviert und in *sites-available* neue virtuelle Hosts angelegt und mit Hilfe von `a2ensite` aktiviert. In den VHosts haben Sie angegeben, in welche Dateien für den jeweiligen Host die Log-Informationen geschrieben werden sollen. Doch auch bevor Sie diese Anweisungen eingetragen haben, hat Apache fleißig Logs geschrieben. Woher kommt das?

Das liegt daran, dass es in der Hauptkonfiguration bereits `ErrorLog`- und `CustomLog`-Anweisungen gibt oder gab.

Übung 33:

1. Öffnen Sie die Datei `/etc/apache2/apache2.conf` und suchen Sie dort die `ErrorLog`-Anweisung.
2. Öffnen Sie die Datei `/etc/apache2/sites-available/000-default` und suchen dort die `CustomLog`-Anweisung. Ist diese Anweisung immer noch aktiv?

In der Datei `apache2.conf` finden Sie zum Beispiel eine `ErrorLog`-Anweisung. Da sich diese in der globalen Konfiguration befindet, gilt sie automatisch für alle virtuellen Hosts, die Sie konfigurieren werden es sei denn, Sie überschreiben die globale Anweisung mit einer Anweisung in der VHost-Konfiguration.

Konfigurationsanweisungen vererben sich also automatisch an die untergeordneten Elemente der Konfiguration weiter. Solange Sie nichts überschreiben, gelten die globalen Einstellungen. Es gibt mehrere Ebenen, auf denen Sie Einstellungen vornehmen können.

11.1.1 Globale Konfiguration

Einstellungen, die Sie in der globalen Konfiguration vornehmen, gelten für alle weiteren Bereiche der Konfiguration. Hier sollten Sie Standardwerte festlegen, die in den meisten Fällen akzeptabel sind.

11.1.2 VirtualHost-Konfiguration

Innerhalb des `VirtualHost`-Containers können Sie Einstellungen für einen einzelnen VHost vornehmen. Was Sie dort konfigurieren, hat keine Auswirkung auf andere virtuelle Hosts.

11.1.3 Konfiguration auf Verzeichnisebene

Sie können viele Einstellungen auch nur für ein einzelnes Verzeichnis vornehmen. So können Sie für ein bestimmtes Verzeichnis einen vom Standard abweichenden `DirectoryIndex` festlegen, falls dies notwendig sein sollte. Diese Anweisung gilt dann für dieses Verzeichnis und alle Unterverzeichnisse es sei denn, weiter unten wird diese Anweisung erneut überschrieben. Auch hier greift also das Apache-Vererbungskonzept.

11.1.4 Konfiguration für einzelne Dateien

Schlussendlich können Sie sogar für einzelne Dateien Sonderregeln festlegen. Sie können zum Beispiel für bestimmte Besucher den Zugriff auf eine einzelne Datei verbieten oder auf alle Dateien mit diesem Namen.

11.1.5 Anwendungsbereiche einzelner Anweisungen

Nicht jede Konfigurationsanweisung kann in jedem Zusammenhang verwendet werden. Es ist absolut sinnvoll, die Ports, auf denen Apache lauscht, global für den gesamten Server festzulegen. Weniger Sinn macht es, die Ports für ein einzelnes Verzeichnis zu ändern.

Daher ist festgelegt, in welchem Zusammenhang Einstellungen vorgenommen werden dürfen. Dies ist in der Apache-Dokumentation bei jeder Anweisung angegeben. Öffnen Sie im Browser die Ihnen bereits bekannte URL <http://httpd.apache.org/docs/2.4/en/mod/directives.html> und sehen Sie sich die folgenden Anweisungen an:

- `Listen`
- `ErrorLog`
- `ErrorDocument`

Bei jeder Anweisung finden Sie direkt zu Beginn der Dokumentation eine Box mit den wichtigsten Daten. Bei `Listen` sieht der Inhalt zum Beispiel so aus:

Description:	IP addresses and ports that the server listens to
Syntax:	<code>Listen [IP-address:]portnumber [protocol]</code>
Context:	server config
Status:	MPM
Module:	event, worker, prefork, mpm_winnt, mpm_netware, mpmt_os2
Compatibility:	The protocol argument was added in 2.1.5

Codebeispiel 21 Listen-Anweisung

Uns interessiert hier vor allem der *Context*. Dieser legt fest, wo Sie diese Anweisung verwenden dürfen. Bei `Listen` finden Sie `server config`, was bedeutet, Sie dürfen diese Anweisung nur global für den ganzen Server setzen.

Bei `ErrorLog` finden Sie beim `Context` `Server config`, `virtual host`, was bedeutet, Sie dürfen diese Anweisung sowohl global als auch in einem virtuellen Host verwenden. Sie können diese Anweisung allerdings nicht für einzelne Verzeichnisse überschreiben.

Die Anweisung `ErrorDocument` ist von den dreien am flexibelsten einsetzbar:

```
Context: Server config, virtual host, directory, .htaccess
```

Sie dürfen diese Anweisung überall einsetzen, global, pro VHost und für jedes Verzeichnis (**engl.** `directory`) ebenfalls. Was die Option `.htaccess` bedeutet, werden Sie gleich erfahren, also ein wenig Geduld.

Achten Sie darauf, dass Sie Anweisungen nur dort einsetzen, wo sie auch gestattet sind. Ziehen Sie regelmäßig die Apache-Dokumentation zu Rate.



11.2 Directory-Container

Sie wissen nun, für welche Bereiche Sie prinzipiell Apache-Einstellungen setzen können. Doch wie können Sie eine Einstellung einem bestimmten Verzeichnis zuweisen?

Zu diesem Zweck gibt es den `Directory`-Container. Dieser sieht dem `VirtualHost`-Container sehr ähnlich und arbeitet auch nach dem gleichen Prinzip. Alle Anweisungen zwischen dem öffnenden und schließenden Tag gelten für das angegebene Verzeichnis und alle Unterverzeichnisse.

Beispiel

```
1 <Directory /var/www/www.example.com/htdocs/images>
2   ErrorDocument 404 /image_not_found.html
3 </Directory>
```

Codebeispiel 22 *Directory-Anweisung*

Die `Directory`-Anweisung erwartet als Argument den absoluten Dateisystem-Pfad, für den die enthaltenen Anweisungen gelten sollen.

Das Beispiel besagt, dass die Webseite http://www.example.com/image_not_found.html ausgeliefert wird, wenn im Ordner `/var/www/www.example.com/htdocs/images` ein 404-Fehlercode ausgelöst wird. Dabei ist es egal, welche Fehler-Dokumente global, im aktuellen VHost oder in darüberliegenden Verzeichnissen konfiguriert wurden.

Übung 34:

1. Legen Sie für beide VHosts `Directory`-Container für die Unterordner `images` und `javascript` an. Geben Sie jedem Ordner ein eigenes Fehler-Dokument für den Fehlercode 404.
2. Legen Sie die Fehler-Webseiten an und testen Sie, ob sie tatsächlich im Browser angezeigt werden, wenn ein 404-Fehlercode erzeugt wird.

11.3 Location-Container

Die `Location`-Anweisung ist mit `Directory` eng verwandt. Der einzige Unterschied ist, dass sie als Argument den Pfad-Teil einer URL erwartet, keinen Dateisystem-Pfad. Mit `Location` können Sie also Regeln für bestimmte URLs und darunterliegende Ebenen festlegen.

Beispiel

```
1 <Location /images>
2     ErrorDocument 404 /image_not_found.html
3 </Location>
```

Codebeispiel 23 Location-Anweisung

Wenn Sie das Beispiel genauer betrachten, sehen Sie, dass dieses und das letzte Beispiel (für `Directory`) exakt den gleichen Effekt haben. Da der URL-Pfad, dem das `DocumentRoot` vorangestellt wird, dem Dateisystem-Pfad entspricht, haben wir wieder folgende »Rechnung«:

```
/var/www/www.example.com/htdocs + /images = /var/www/www.example.com/htdocs/
images
```

also

```
DocumentRoot + Location-Anweisung = Directory-Anweisung
```

Welchen der beiden Wege Sie bevorzugt verwenden, bleibt Ihnen überlassen. Sie sollten nur darauf achten, beide Anweisungen nicht wild zu mischen, da Sie sonst leicht die Übersicht verlieren, welche Anweisung gerade gilt. Bis auf einige wenige, besondere Fälle sind beide tatsächlich austauschbar.³⁹

Übung 35:

Schreiben Sie die `Directory`-Anweisungen aus Übung 34 in `Location`-Anweisungen um. Achten Sie darauf, die Pfad-Argumente korrekt zu setzen.

11.4 Files-Container

Die Anweisung `Files` ist nach `VirtualHost`, `Directory` und `Location` der vierte Container. Diesem können Sie als Argument einen Dateinamen übergeben, für den die enthaltenen Regeln gelten sollen. Das Interessante ist: Es ist egal, wo genau sich diese Datei befindet, es geht wirklich nur um den Dateinamen.

39. `Location` benötigen Sie unbedingt, wenn URLs nicht auf dem Dateisystem existieren. Viele modernen Webframeworks und CMS-Systeme (*Zend Framework, Symfony, Rails, Django, Drupal...*) verwenden generierte URLs, denen keine echte Datei bzw. Verzeichnis entspricht. Wenn Sie auf derartige URLs Sonderregeln anwenden wollen, benötigen Sie `Location`, nicht `Directory`.

Beispiel

```
<Files .htaccess>
  Deny from All
</Files>
```

Codebeispiel 24 Files-Anweisung

Für dieses Beispiel musste ich ein klein wenig vorgreifen. Die Einstellung `Deny from All` bewirkt, dass der Zugriff für jeden untersagt ist. Also sorgt der Code im Beispiel dafür, dass für alle der Zugriff auf alle Dateien namens `.htaccess` gesperrt ist. Sie können also Dateien mit diesem Namen nicht über den Browser abrufen.

Übung 36:

1. Erstellen Sie in einem Ihrer VHosts eine `Files`-Anweisung, die für alle Dateien mit dem Namen `secret.html` gilt. Der `Files`-Container soll die Anweisung `Deny from All` enthalten, also für alle Browser den Zugriff auf Dateien mit diesem Namen sperren.
2. Legen Sie im `DocumentRoot` dieses VHosts in verschiedenen Verzeichnissen mehrere Dateien namens `secret.html` an. Versuchen Sie, diese über den Browser zu öffnen.

11.5 .htaccess-Dateien

11.5.1 Konzept

Solange Sie einen Apache-Server nur für sich selbst konfigurieren, sprich Sie selbst Ihr Kunde sind, werden Sie `.htaccess`-Dateien nicht benötigen. Sobald Sie allerdings für mehrere Personen Webseiten auf Ihrem Server hosten, werden Ihre Kunden regelmäßig Änderungen an der Serverkonfiguration beantragen: Sei es, dass die Kunden für bestimmte Unterordner einen Passwortschutz möchten, dass der `DirectoryIndex` geändert werden soll oder dass bestimmte URL-Umleitungen gesetzt werden sollen. Alle diese Anforderungen werden für Sie nicht schwer zu lösen sein, aber sie werden Zeit kosten.

Daher gibt es für Apache die Möglichkeit, bestimmte Teile der Serverkonfiguration aus der eigentlichen Apache-Konfiguration auszulagern und in das `DocumentRoot` des VHosts, oder auch in Unterordner, zu verschieben. Da diese Verzeichnisse im Zugriff des Kunden liegen, kann dieser nun viele Einstellungen selbst vornehmen zumindest wenn Sie einen Kunden mit Apache-Erfahrung haben, was auf die meisten PHP-Entwickler aber zutreffen sollte.

Der Kunde muss nun in dem Verzeichnis, in dem die Einstellungen aktiv werden sollen, eine Datei mit dem Namen `.htaccess`⁴⁰ anlegen und dort die gewünschten Anweisungen eintragen. Diese haben dann dieselbe Wirkung, als wenn Sie selbst sie in einen `Directory`-Container in der VHost-Konfiguration eingetragen hätten. Sie müssen den Verzeichnispfad von Hand ein-

40. Achtung: Da diese Datei mit einem Punkt beginnt, gilt sie für Unix-Systeme als versteckt! Den Dateinamen könnten Sie zwar prinzipiell in der Apache-Konfiguration ändern, davon rate ich Ihnen aber stark ab. Dieser Name hat sich durchgesetzt und ist mittlerweile als Standard anzusehen.

tragen, bei `.htaccess`-Dateien wird das betroffene Verzeichnis automatisch bestimmt es ist einfach das Verzeichnis, in dem die Datei liegt.

11.5.2 AllowOverride

Damit Ihre Kunden nicht jede beliebige Anweisung setzen dürfen, was unter Umständen ein Sicherheitsproblem sein könnte, können Sie bestimmen, welche Anweisungen gestattet sind. Verwendet der Kunde eine unerlaubte, erzeugt Apache einen Fehler mit dem Code 500, also einen Serverfehler. Dies gilt jedoch nur für den Ordner, der die `.htaccess`-Datei enthält, was ein weiterer Vorteil dieser Art der Konfiguration ist. Diese Dateien können immer nur die Konfiguration des betroffenen Verzeichnisses beeinflussen, und niemals andere Verzeichnisse oder Hosts in Mitleidenschaft ziehen.

Die Anweisung, mit der Sie festlegen, welche Anweisungen verwendet werden dürfen, heißt `AllowOverride`. Diese Anweisung dürfen Sie ausschließlich innerhalb von `Directory`-Containern verwenden.

Als Argumente können Sie fünf Gruppen von Anweisungen angeben, die entweder erlaubt oder verboten sein sollen. Für die vollständige Liste, welche Anweisungen zu welcher Gruppe gehören, möchte ich Sie wieder auf die Apache-Dokumentation verweisen:

<http://httpd.apache.org/docs/2.4/en/mod/core.html#allowoverride>

Folgende Anweisungsgruppen existieren:

- ▶ `AuthConfig`: Diese Gruppe enthält alle Anweisungen, die mit der Authentifizierung von Benutzern zu tun haben, also das Prüfen von Passwörtern, Gruppenverwaltung oder die Konfiguration der Benutzerdatenbank.
- ▶ `FileInfo`: Dort finden Sie Anweisungen, welche die Dokumenttypen verwalten, also zum Beispiel die Sprache festlegen oder per `ErrorDocument` Fehlerseiten konfigurieren.
- ▶ `Indexes`: Diese Gruppe enthält Anweisungen, mit denen Sie das Verhalten der Index-Seiten konfigurieren können. Dort finden Sie zum Beispiel `DirectoryIndex`.
- ▶ `Limit`: Mit `Limit` werden wir uns in [Lektion 12](#) »Zugriffskontrolle« beschäftigen. Hier finden Sie Anweisungen zur Zugriffskontrolle, also wer die Seiten sehen darf und wer nicht.
- ▶ `Options`: Dort können Sie festlegen, ob in der `.htaccess`-Datei die `Options`-Anweisung verwendet werden darf.
- ▶ `None`: Eigentlich ist `None` gar keine Gruppe. Damit deaktivieren Sie ausdrücklich alle `AllowOverride` und verbieten die Verwendung von `.htaccess`-Dateien. Sie wird benötigt, wenn in einem übergeordneten Verzeichnis die Verwendung gestattet wurde. Dies ist die Standardeinstellung für Apache 2.4.
- ▶ `All`: Ist ebenfalls keine Gruppe. Erlaubt alle Anweisungen, die im `.htaccess`-Kontext erlaubt sind. Dies ist die Standardeinstellung für Apache 2.2.

Beispiel

```
1 <Directory /var/www/www.example.com/htdocs>
2     AllowOverride AuthConfig Limit
3 </Directory>
```

Codebeispiel 25 `AllowOverride`

In diesem Beispiel wurde für den gesamten VHost www.example.com⁴¹ die Verwendung von `.htaccess`-Dateien erlaubt, allerdings nur für die Anweisungsgruppen `AuthConfig` und `Limit`. Wir mussten einen `Directory`-Container verwenden, da `AllowOverride` nur dort erlaubt ist. Mit diesem Trick, das `Directory` auf das `DocumentRoot` zeigen zu lassen, können Sie `AllowOverride` trotzdem VHost-weit einsetzen.

Beispiel

```
1 <Directory /var/www/www.example.de/htdocs>
2     AllowOverride All
3 </Directory>
```

Codebeispiel 26 `AllowOverride`

Für den virtuellen Host www.example.de⁴² wurde die Verwendung von allen Direktiven gestattet, die im `.htaccess`-Kontext erlaubt sind.

Übung 37:

1. Erstellen Sie für beide VHosts einen `Directory`-Container für das jeweilige `DocumentRoot`. Setzen Sie dort eine `AllowOverride`-Anweisung.
2. Der erste VHost soll `Limit` aktiviert haben, der zweite `FileInfo` und `Indexes`.

11.5.3 .htaccess-Dateien editieren

Kommen wir zu guter Letzt zum eigentlichen Erstellen von `.htaccess`-Dateien. Dazu müssen Sie eine Datei namens `.htaccess` im gewünschten Verzeichnis erstellen und dort die Anweisungen eintragen.

Beispiel

```
1 DirectoryIndex test.html
2 Options -Indexes
3 ErrorDocument 404 /fehler.html
```

Codebeispiel 29 `/var/www/www.example.com/htdocs/.htaccess`

Nach dem ganzen Aufwand zur Vorbereitung ist das doch reichlich unspektakulär, oder? Aber genau darum geht es; die Kunden sollen so einfach wie möglich selbst Änderungen an der Konfiguration vornehmen können.

Übung 38:

1. Erstellen Sie im `DocumentRoot` Ihrer beiden VHosts `.htaccess`-Dateien. Tragen Sie dort verschiedene Anweisungen ein, um die Funktionsweise zu testen.

41. <http://www.example.com>

42. <http://www.example.de>

2. Achten Sie darauf, nur Anweisungen zu verwenden, die für den Gebrauch in *.htaccess*-Dateien vorgesehen sind. Dies sehen Sie in der Apache-Dokumentation, wenn unter *Context* `.htaccess` aufgeführt ist.

11.6 Zusammenfassung

Apache ermöglicht es, Einstellungen sowohl global als auch nur für bestimmte Bereiche vorzunehmen. Dabei verwendet er die Regel, dass immer die spezifischste Anweisung für einen Bereich gilt. Auf diese Weise können Sie bequem Standardwerte festlegen, diese aber jederzeit überschreiben, wenn es nötig ist.

Sie können dank *.htaccess*-Dateien sogar Teile der Konfiguration Ihren Kunden überlassen und müssen dabei trotzdem nicht die Kontrolle über den Server abgeben.

11.7 Testen Sie Ihr Wissen

1. Wo können Sie nachsehen, für welche Konfigurations-Bereiche eine Apache-Anweisung verwendet werden darf?
2. Was ist der Unterschied zwischen `Directory` und `Location`?
3. Welche Anweisung müssen Sie setzen, um die Verwendung von *.htaccess*-Dateien komplett zu verbieten?