



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 1.3.0 vom 28.02.2020

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm

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	14
2.2.1 PHP-Backports von packages.sury.org	15
2.2.2 MySQL oder doch MariaDB?	16
2.2.3 Installation	17
2.2.4 Testen der Installation	18
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	21
3.4 CMS Made Simple	21
3.4.1 Herunterladen und installieren der Anwendungs-Dateien	22
3.4.2 Anlegen der Datenbank	22
3.4.3 Konfiguration über das Webinterface	23
3.5 Zusammenfassung	26
3.6 Testen Sie Ihr Wissen	26
4 PHP-Administration	28
4.1 PHP für Kommandozeile und Webserver	28
4.1.1 Installation von php7.2-cli	28
4.1.2 PHP-Skripte als Shellkommandos	28
4.2 Grundkonfiguration	29
4.2.1 Konfiguration für Webserver und Kommandozeile	29
4.2.2 Allgemeine Einstellungen	29
4.2.3 Ressourcenverwaltung	31
4.3 Konfiguration von PHP-Erweiterungen	32
4.3.1 Konzept	32
4.3.2 Session	33
4.3.3 Installation von PHP-Erweiterungen über apt	35

4.4	Zusammenfassung	37
4.5	Testen Sie Ihr Wissen	37
5	Einführung in MySQL	39
5.1	Verzeichnisstruktur von MySQL auf Debian	39
5.1.1	/etc/mysql	39
5.1.2	/var/lib/mysql	39
5.2	MySQL-Kommandozeilenclient	40
5.2.1	Verbindung zum Datenbankserver aufbauen	40
5.2.2	Befehle an MySQL senden	41
5.3	SQL für Administratoren	42
5.3.1	Konzept	42
5.3.2	Datenbanken erzeugen und löschen	42
5.3.3	Auslesen von Informationen mit SHOW	43
5.3.4	Datensätze auslesen	46
5.4	Zusammenfassung	47
5.5	Testen Sie Ihr Wissen	47
6	MySQL: Benutzer und Rechte	49
6.1	Benutzer und Rechte in MySQL	49
6.1.1	Die Datenbank mysql	49
6.2	Benutzerverwaltung	50
6.2.1	Passwort ändern	50
6.2.2	Das auth_socket Authentication Plugin	50
6.2.3	Benutzer löschen	52
6.2.4	Benutzer anlegen	52
6.3	Rechteverwaltung	53
6.3.1	Rechte ansehen	53
6.3.2	Rechte gewähren	54
6.3.3	Rechte nehmen	57
6.4	Zusammenfassung	58
6.5	Testen Sie Ihr Wissen	58
7	MySQL: Backup und Recovery	60
7.1	MySQL-Datenbanken reparieren	60
7.1.1	Beschädigte Tabellen	60
7.1.2	mysqlcheck	60
7.1.3	mysqlcheck --repair	60
7.2	Backup von MySQL-Datenbanken	62
7.2.1	Notwendigkeit	62
7.2.2	mysqldump	62
7.2.3	Backups mit mysqldump automatisieren	63
7.2.4	Replikation	64
7.3	Einspielen von Sicherungen	64
7.3.1	Gründe für eine komplette Wiederherstellung	64
7.3.2	Einspielen von Sicherungen	65
7.3.3	Wiederherstellung einer Datenbank	65
7.3.4	Wiederherstellung des kompletten MySQL-Servers	66
7.4	Zusammenfassung	68
7.5	Testen Sie Ihr Wissen	68

8	Von MySQL zu MariaDB	70
8.1	Hintergrund	70
8.2	MySQL und MySQL-Quellen entfernen	70
8.3	MariaDB installieren	70
8.4	Die Datenbank für CMSMS einspielen	71
8.5	Unterschiede zwischen MySQL und MariaDB	71
8.6	Zusammenfassung	72
9	Das HTTP-Protokoll	73
9.1	Aufbau des HTTP-Protokolls	73
9.1.1	Beispiel für eine Anfrage über HTTP	73
9.1.2	Versionen des HTTP-Protokolls	73
9.2	HTTP-Request	75
9.2.1	Request-Methoden	76
9.2.2	Client-Request-Header	76
9.3	HTTP-Response	77
9.3.1	Server-Response-Header	78
9.4	Zusammenfassung	79
9.5	Testen Sie Ihr Wissen	79
10	Konfiguration von Apache	80
10.1	Struktur der Apache-Konfiguration	80
10.1.1	apache2.conf	80
10.1.2	ports.conf	80
10.1.3	envvars	80
10.1.4	conf-available, conf-enabled	80
10.1.5	mods-available, mods-enabled	81
10.1.6	sites-available, sites-enabled	81
10.2	Apache-Dokumentation	81
10.3	Grundkonfiguration	81
10.4	Apache-Module	82
10.4.1	Konzept	82
10.4.2	Aktivierung eines Moduls	82
10.4.3	Deaktivierung eines Moduls	84
10.5	Zusammenfassung	84
10.6	Testen Sie Ihr Wissen	84
11	Virtuelle Hosts	86
11.1	Virtuelle Hosts	86
11.1.1	Konzept	86
11.1.2	Typen von virtuellen Hosts	86
11.2	Konfiguration namensbasierter virtueller Hosts	88
11.2.1	Voraussetzungen	88
11.2.2	NameVirtualHost-Direktive	89
11.2.3	VirtualHost-Container	90
11.2.4	Aktivieren und Deaktivieren von virtuellen Hosts	93
11.2.5	Optionale Direktiven	94
11.2.6	Options	97
11.3	Zusammenfassung	98
11.4	Testen Sie Ihr Wissen	99

12	Konfigurations-Bereiche	100
12.1	Geltungsbereich von Apache-Anweisungen	100
12.1.1	Globale Konfiguration	100
12.1.2	VirtualHost-Konfiguration	100
12.1.3	Konfiguration auf Verzeichnisebene	100
12.1.4	Konfiguration für einzelne Dateien	101
12.1.5	Anwendungsbereiche einzelner Anweisungen	101
12.2	Directory-Container	102
12.3	Location-Container	102
12.4	Files-Container	103
12.5	.htaccess-Dateien	103
12.5.1	Konzept	103
12.5.2	AllowOverride	104
12.5.3	.htaccess-Dateien editieren	105
12.6	Zusammenfassung	106
12.7	Testen Sie Ihr Wissen	107
13	Zugriffskontrolle	108
13.1	Anforderungen	108
13.2	Hostbasierte Zugriffsbeschränkung	108
13.2.1	Konzept	108
13.2.2	Allow und Deny	109
13.2.3	Order	110
13.3	Require	111
13.3.1	RequireAny-Container	112
13.3.2	RequireAll-Container	113
13.4	Beschränkung durch Authentifizierung	113
13.4.1	Konzept	113
13.4.2	Konfiguration	113
13.4.3	Erstellen der Passwort-Datei	115
13.5	Zusammenfassung	116
13.6	Testen Sie Ihr Wissen	116
14	URL-Manipulation	119
14.1	Zusammenhang zwischen URL und Dateipfad	119
14.2	Alias	120
14.3	Redirect	120
14.4	Reguläre Ausdrücke	122
14.4.1	Einführung	122
14.5	URL-Manipulation mit mod_rewrite	123
14.5.1	Konzept und Installation	123
14.5.2	RewriteEngine	123
14.5.3	RewriteRule	123
14.5.4	RewriteCond	125
14.6	Zusammenfassung	127
14.7	Testen Sie Ihr Wissen	128
15	Virtuelle Hosts mit SSL/TLS	129
15.1	HTTPS	129
15.2	Erstellen von SSL-Zertifikaten	129
15.3	Konfiguration von Apache	131
15.3.1	HTTPS-Port	131

15.3.2	mod_ssl	132
15.3.3	VHost-Konfiguration	132
15.3.4	SSL und IP-Adressen	134
15.4	Zusammenfassung	135
15.5	Testen Sie Ihr Wissen	135
16	Log-Analyse	137
16.1	Anforderungen	137
16.2	Informationen aus Logdateien	137
16.3	Webalizer	137
16.4	Piwik	138
16.5	AWStats	138
16.6	Konfiguration von AWStats	138
16.6.1	Installation des Programms	138
16.6.2	Ausführen von CGI-Skripten	138
16.6.3	Konfiguration von AWStats	140
16.6.4	Analyse der Logdatei	141
16.6.5	Regelmäßige Aktualisierung der Analysedaten	141
16.7	Schützen der Statistikseite	142
16.8	Zusammenfassung	142
16.9	Testen Sie Ihr Wissen	142
17	Sicherheit	144
17.1	Sicherheit in Apache	144
17.1.1	Listen	144
17.1.2	Dateisystem	145
17.1.3	ServerTokens	147
17.1.4	Includes	148
17.1.5	CGI-Skripte	149
17.2	Zusammenfassung	149
17.3	Testen Sie Ihr Wissen	149
	Lösungen der Übungsaufgaben	151
	Lösungen der Wissensfragen	165
	Index	175

12 Konfigurations-Bereiche

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.

12.1 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 31:

- 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.conf` 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.

12.1.1 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.

12.1.2 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.

12.1.3 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.

12.1.4 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.

12.1.5 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 20 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.



12.2 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 21 *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 32:

- 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.

12.3 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 22 *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.³⁶

Übung 33:

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

12.4 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.

Beispiel

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

Codebeispiel 23 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 34:

- 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.

12.5 .htaccess-Dateien

12.5.1 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 regelmäßige Änderungen an der Serverkonfiguration beantragen: Sei es, dass die Kunden für bestimmte Unterordner einen Passwortschutz möchten, dass der

36. `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`.

`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` anlegen.



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.

In der `.htaccess`-Datei kann er nun 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 eintragen, bei `.htaccess`-Dateien wird das betroffene Verzeichnis automatisch bestimmt es ist einfach das Verzeichnis, in dem die Datei liegt.

12.5.2 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 13 »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 Standareinstellung 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 24 *AllowOverride*

In diesem Beispiel wurde für den gesamten VHost www.example.com³⁷ 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 25 *AllowOverride*

Für den virtuellen Host www.example.de³⁸ wurde die Verwendung von allen Direktiven gestattet, die im **.htaccess**-Kontext erlaubt sind.

Übung 35:

- 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**.

12.5.3 .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 28 */var/www/www.example.com/htdocs/.htaccess*

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

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

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 36:

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

12.6 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.

12.7 Testen Sie Ihr Wissen

1. Für das Verzeichnis `/post` Ihrer Website `www.example.com` möchten Sie erreichen, dass bei Aufruf einer nicht existierenden Seite aus dem Webserver-Verzeichnis `www.example.com/posts` die Seite `www.example.com/posts/fehler.html` ausgeliefert wird. Ihre Konfiguration sieht bisher wie folgt aus:

```
<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot /var/vhosts/www.example.com/htdocs
    ServerAdmin webmaster@example.com
</VirtualHost>
```

Mit welcher Konfiguration können Sie Ihr Ziel erreichen?

Bitte ankreuzen:

- ☐

```
<Directory /var/www/www.example.com/htdocs/posts>
    ErrorDocument fehler.html
</Directory>
```
- ☐

```
<Location /var/www/www.example.com/htdocs/posts>
    Error 200 /fehler.html
</Location>
```
- ☐

```
<Location /posts>
    ErrorDocument 404 /posts/fehler.html
</Location>
```
- ☐

```
<Directory /posts>
    FileNotFound 404 fehler.html
</Directory>
```

2. Welche Aussagen zu den Containern `Directory` und `Location` in der dezentralen Datei zur Verwaltung der Konfiguration eines Apache-Webservers sind zutreffend?

Bitte ankreuzen:

- ☐ `Directory` bezieht sich auf den Pfad im Dateisystem.
- ☐ `Directory` bezieht sich auf den Pfadteil der URL.
- ☐ `Location` bezieht sich auf den Pfadteil der URL.
- ☐ `Location` bezieht sich auf den Pfad im Dateisystem.

3. Sie wollen verhindern, dass ein Kunde mit `.htaccess` Dateien die Apache-Konfiguration anpassen kann. Welche Einstellung löst das Problem?

Bitte ankreuzen:

- ☐ `Deny All .htaccess`
- ☐

```
<Files .htaccess>
    Deny from All
</Files>
```
- ☐ `AllowOverride None`
- ☐ `DenyOverride All`