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

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:             Listen [IP-address:]portnumber [protocol]
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

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

```
<Location /images>
    ErrorDocument 404 /image_not_found.html
</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 *.htacces* 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 regelmäßig 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

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

Codebeispiel 24 *AllowOverride*

In diesem Beispiel wurde für den gesamten VHost www.example.com[37] 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

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

Codebeispiel 25 *AllowOverride*

Für den virtuellen Host www.example.de[38] 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

```
DirectoryIndex test.html
Options -Indexes
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 unspektatulär, 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*-Datein 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
   ```