Der Apache Webserver

ID:	**TT-NC7**
Dokumenten URL:	**http://docs.tx7.de/TT-NC7**
Autor:	**Tom Gries <tom@tx7.de>**
Version:	**5.0.1 vom 13.07.2017**

>> Themen

- Aufgabe und Funktion eines Webservers
- Virtuelle Hosts und das DocumentRoot
- Die HTTP Status-Codes
- Die .htaccess-Datei - Einstieg
- Die .htaccess-Datei - Authentifizierung
- Die .htaccess-Datei - Redirects
- Referenzen

>> Aufgabe und Funktion eines Webservers

1. Statische Dokumente ausliefern:

Die ursprüngliche Aufgabe eines Webservers war simpel:

- ⇒ Eine angeforderte URL in einen Dateinamen zu übersetzen.
- ⇒ Die Datei suchen (und öffnen).
- ⇒ Die Datei als Antwort zurück zu senden.

Statische Dokumente werden genau so übertragen wie sie sind (unverändert).

2. Dynamische Dokumente ausliefern:

Die statische Grundfunktion wurde recht früh um dynamische Funktionalitäten erweitert. Hierzu werden die Dokumente vor der Auslieferung an ein Programm übergeben. Dieses Programm führt die enthaltenen Anweisungen aus, stellt das Ergebnis zusammen und übergibt es wieder an den Webserver.

⇒ Der Webserver übernimmt dann wieder die Auslieferung an den Client.

>> Aufgabe und Funktion eines Webservers

3. Weitere Funktionen:

Neben den beiden zuvor genannten Funktionen kann ein Webserver heute noch weitere Aufgaben übernehmen. Für den Apache Webserver sind dies zum Beispiel:

- ⇒ Authentifizierung und Zugriffsbeschränkung
- ⇒ Verschlüsselung der Übertragung (HTTPS)
- ⇒ Weiterleitungen
- ⇒ … und vieles mehr.

>> Virtuelle Hosts und das DocumentRoot

Der Apache kann - wie die meisten Webserver - mehrere Hosts/Domains gleichzeitig verwalten. Diese Hosts nennen sich dann virtuelle Hosts. So können auf einem physischen Server mit nur einer einzigen Apache Instanz zum Beispiel die Domains 7xp.de, tx7.de und docs.tx7.de gehostet werden. Dies ist seit HTTP 1.1 möglich. Dazu wird im HTTP Header der Host, für den die Anfrage gilt, mit angegeben:

```
GET / HTTP/1.1
Host: tx7.de
```

>> Virtuelle Hosts und das DocumentRoot

Wenn man genau hinschaut, erkennt man die einzelnen Bestandteile einer URL (siehe Vorlesung zu den URL's: http://docs.tx7.de/TT-H5R). Das Scheme (http), den Host (tx7.de) und den URL-Path (/). Zusammengesetzt ergibt das

```
http://tx7.de/
```

Der URL-Path / wird auf dem Webserver mit einer der konfigurierten Startseiten ersetzt. Dazu später mehr.

>> Virtuelle Hosts und das DocumentRoot

Um Dokumente - statisch oder dynamisch - ausliefern zu können, wird für jeden virtuellen Host ein sogenanntes DocumentRoot im Dateisystem auf dem Webserver festgelegt. Dies ist das Startverzeichnis der virtuellen Hosts und entspricht dem ersten / im URL-Path.

`http://tx7.de/`

Entspricht dem DocumentRoot

>> Virtuelle Hosts und das DocumentRoot

Beispiel:

Für die Domain tx7.de legen wir das Verzeichnis **/srv/www/tx7.de/htdocs** als DocumentRoot in der Apache Konfigurationsdatei fest. In diesem Verzeichnis erstellen wir dann ein weiteres Verzeichnis "images" und kopieren dort das Bild logo.png hinein.

Wenn wir jetzt http://tx7.de/images/logo.png aufrufen, sehen wir genau dieses Bild. Anm.: Der Host tx7.de muss hierfür natürlich korrekt im DNS eingetragen sein.

>> Die HTTP Status-Codes

Folgende HTTP Status-Codes Bereiche sind definiert:

1xx	Informationen
2xx	Erfolgreiche Operationen
3xx	Umleitungen
4xx	Client-Fehler
5xx	Server-Fehler

>> Die HTTP Status-Codes

Die am häufigsten vorkommenden Status-Codes sind:

200	OK
301	Moved Permanently
302	Moved Temporarily
401	Unauthorized
403	Forbidden
404	Not Found
500	Server-Fehler

>> Die .htaccess-Datei - Einstieg

Bei der .htaccess handelt es sich um eine Apache Konfigurationsdatei, in der der User (Website-Entwickler) Konfigurationen vornehmen kann. Diese müssen zuvor vom Administrator des Servers freigegeben worden sein. Dadurch können auch Voreinstellungen individuell angepasst (also überschrieben) werden.

Sie liegt im DocumentRoot oder darunterliegenden Verzeichnissen und es kann mehrere .htaccess Dateien geben. In diesem Fall "addieren" sich die Einträge. Falls eine Direktive schon gesetzt war, gilt immer die Letzte.

>> Die .htaccess-Datei - Einstieg

Beispiel:

Als DocumentRoot für tx7.de ist **/srv/www/tx7.de/htdocs** konfiguriert. In diesem DocumentRoot befindet sich ein weiteres Verzeichnis **.admin**.

In beiden Verzeichnissen befindet sich eine .htaccess-Datei, in denen festgelegt wird, ob der Inhalt des Verzeichnisses angezeigt wird oder nicht. In der .htaccess-Datei im **DocumentRoot** ist die Anzeige des Inhalts disabled - in der .htaccess im Verzeichnis **.admin** ist es enabled.

>> Die .htaccess-Datei - Einstieg

Die .htaccess-Datei genießt einen besonderen Schutz. In der Hauptkonfigurationsdatei des Apache Web-Servers ist festgelegt, dass Dateien, die mit **.ht** Anfangen, nicht ausgeliefert werden.

Diese ist extrem Wichtig für geschützte Seiten.

Es ist möglich, den Namen der .htaccess-Datei in der Apache Hauptkonfigurationsdatei zu ändern. Dies sollte man aber nur tun, wenn man ganz genau weiß, was für Auswirkungen dies hat.

>> Die .htaccess-Datei - Einstieg

Einzelne Einstellungen (Zeilen) in der .htaccess-Datei werden Direktiven genannt. Häufig genutzte Direktiven werden wir uns im nachfolgenden anschauen.

Es gibt Direktiven, die grundsätzlich verfügbar sind (core), andere stehen erst durch die Aktivierung von Modulen zur Verfügung (z. B. mod_rewrite).

Grundsätzlich ist die Groß-/Kleinschreibung der Direktiven zu beachten, da es sonst zu einem Fehler (HTTP Status-Code 500) kommen kann.

Options [+|-]Option [[+|-]Option] …:

Die Direktive **Options** steuert, welche Funktionen in einem bestimmten Verzeichnis verfügbar sind. Es handelt sich hierbei um eine Aufzählung mehrerer Optionen. Ohne +|- werden **sämtliche** vorherigen Optionen überschrieben. Normalerweise will man aber nur **eine** Option hinzufügen oder entfernen. Dann ist ein + oder - notwendig. Folgende Direktive zeigt beispielsweise den Inhalt eines Verzeichnisses zusätzlich zu den bereits bestehenden an:

```
Options +Indexes
```

>> Die .htaccess-Datei - Einstieg

Aufgabe 1:

Beim Aufruf von **http://virtualtux.de/~tux-nnn** wird der Inhalt des Verzeichnisses angezeigt. In der .htacess-Datei die Direktive so setzen, dass der Inhalt nicht mehr angezeigt wird.

Anm.: Eine index.html oder index.php darf zum Testen nicht vorhanden sein.

>> Die .htaccess-Datei - Einstieg

DirectoryIndex:

Die Direktive **DirectoryIndex** legt fest, welche Seite oder Seiten automatisch angezeigt werden sollen. Meistens sind Seitennamen wie index.html oder index.php voreingestellt. Es können mehrere Seitennamen angegeben werden. Die Erste Datei, die gefunden wird, wird ausgeliefert.

```
DirectoryIndex wartung.php index.php index.html
```

Aufgabe 2:

Eine Datei start.htm im DocumentRoot mit beliebigem ASCII Inhalt erstellen (oder hochladen). In der .htaccess-Datei eine Direktive erstellen, die diese Datei (start.htm) automatisch **und als erstes** anzeigt.

>> Die .htaccess-Datei - Einstieg

ErrorDocument:

Wenn es bei der Beantwortung einer Anfrage ein Problem gibt, wird ein entsprechender HTTP-Statuscode erzeugt und zurück geliefert. Der Browser entscheidet dann, was angezeigt wird und wie es angezeigt wird.

Es kann aber auch ein eigener Text oder eine beliebige eigene Seite zurück geliefert werden. Zum Beispiel eine Seite, die zum Corperate Design (CD) passt.

>> Die .htaccess-Datei - Einstieg

Nach der Direktive "ErrorDocument" kommt der Status-Code, für den es gelten soll und dann der Text (in Anführungszeichen) oder die URL zu der Datei, die ausgeliefert werden soll.

```
ErrorDocument 404 "Tippfehler?"
```

oder

```
ErrorDocument 404 http://docs.tx7.de/TT-B9H
ErrorDocument 500 http://docs.tx7.de/TT-7TY
```

Aufgabe 3:

Falls noch nicht vorhanden eine .htaccess im eigenen DocumentRoot mit folgendem Inhalt erstellen:

```
ErrorDocument 404 "Tippfehler?"
#ErrorDocument 404 http://docs.tx7.de/TT-B9H
```

Danach eine nichtexistierende Seite aufrufen, z. B.
http://virtualtux.de/~tux-nnn/nzbymf

>> Die .htaccess-Datei - Einstieg

AddType:

Mit der Direktive AddType kann man Dateiendungen (z. B. .html, .php oder andere) einer Anwendung zuordnen.

Die (selbstausgedachten) Endung .tx7 kann mit dieser Direktive wie eine .php Datei vom PHP-Prozessor verarbeitet werden:

```
AddType application/x-httpd-php .tx7
```

>> Die .htaccess-Datei - Zugriffschutz

Zugriffschutz:

Auch ein Zugriffsschutz auf bestimmte Verzeichnisse ist mit der .htaccess-Datei möglich. Hierzu sind ein paar "Vorarbeiten" notwendig. Als Erstes muss eine Datei mit den Usernamen und Passwörtern erstellt werden. Die Datei kann prinzipiell unter beliebigen Namen und an beliebiger Stelle im Verzeichnis gespeichert werden. Es sollten aber zwei Dinge beachtet werden:

Die Datei sollte mit **.ht** beginnen und nach Möglichkeit außerhalb des DocumentRoots gespeichert werden.

>> Die .htaccess-Datei - Zugriffschutz

Wenn diese Datei unter /home/tux-nnn/.htuser gespeichert wird, dann kann die zugehörige .htaccess wie folgt aussehen:

```
AuthType Basic
AuthName "Area 51"
AuthUserFile /home/tux-nnn/.htuser

require valid-user
```

>> Die .htaccess-Datei - Zugriffschutz

Die .htuser kann mit dem Linux Befehl htpasswd erstellt werden. Die genaue Syntax kann unter Linux mit

```
man htpasswd
```

nachgelesen werden. Hier ein

```
htpasswd -c .htuser tux-nnn
```

im **HomeDirectory** (**nicht** DocumentRoot) ausführen.

>> Die .htaccess-Datei - Zugriffschutz

Aufgabe 4:

Im Verzeichnis /home/tux-nnn mit htpasswd die Datei .htuser mit dem eigenen Usernamen (tux-nnn) und einem eigenem Passwort erstellen.

Danach im DocumentRoot public_html ein Verzeichnis **admin** anlegen. Hier eine .htaccess wie auf der Seite zuvor erstellen und testen.

>> Die .htaccess-Datei - Weiterleitungen

Redirect:

Mit der .htaccess-Datei ist es auch möglich, verschiedene Arten von Weiterleitungen festzulegen. Das Thema Weiterleitungen ist sehr umfangreich und es gibt für komplexe Anforderungen das Modul mod_rewrite. Wir werden uns hier nur mit der einfachen Weiterleitung befassen.

Einfach bedeutet hier, dass ein (nicht existierender Pfad) auf eine andere URL gemappt wird.

>> Die .htaccess-Datei - Weiterleitungen

Wenn man nur lokal eine andere Datei ausliefern möchte, dann kann man anstatt eines Redirects auch ein Alias verwenden.

Der wesentliche Unterschied zwischen einem Redirect und einem Alias ist, dass bei einem Redirect der Browser die neue URL mitgeteilt bekommt und sich den Inhalt von dort selber abholen muss. Bei einem Alias wird der Alias-Inhalt direkt an den Browser geliefert. Der Browser "weiß" dadurch nichts von der Ersetzung.

>> Die .htaccess-Datei - Weiterleitungen

Weiterleitungen werden in der .htaccess wie folgt notiert:

```
Redirect /~tux-nnn/go/fb http://facebook.de
```

oder mit explizit gesetztem Status-Code:

```
Redirect 301 /~tux-nnn/dhbw http://dhbw-mannheim.de
```

Der rote Teil ist der (nicht existierende) Pfad und kann sich selber ausgedacht werden. Der blaue Teil ist das Ziel.

>> Die .htaccess-Datei - Weiterleitungen

Aufgabe 5:

Die beiden Weiterleitungen plus zwei eigene in der .htaccess Datei erstellen und testen.

>> Referenzen

Apache 2.4 Dokumentation:

http://docs.tx7.de/TT-UFH

Die .htaccess-Datei (Wikipedia):

http://docs.tx7.de/TT-ERV