



Webdesign

FüUstgSBw
First Edition
HF Weidinger
L Siegerth

Inhaltsverzeichnis

1 Grundlagen	3
1.1 Testseite	4
1.2 Testseite	4
2 HTML	5
2.1 HTM...Was?	5
2.2 Tags	5
2.3 Das Grundgeruest	8
2.3.1 HEAD und BODY	8
2.3.2 Absaetze	8
2.4 DOCTYPE	8
2.5 Sonderzeichen	8
2.6 Zeilenumbrueche	8
3 CSS	10
4 WEBSERVER	11
4.1 HTTP und das Web	11
4.1.1 Webserver und Protokolle	11
4.2 Apache	13
4.2.1 Einfuehrung und Installation	13
4.2.2 Konfiguration	15
4.2.3 Sicherheit	18
4.2.4 Indexes	20
4.2.5 SSI	23
4.2.6 CGI	23

1 Grundlagen

Woher kommt das Internet

Die ersten Anfänge - Das ARPANet

Im amerikanischen Verteidigungsministerium wurde schon seit den 1960er Jahren darüber nachgedacht, wie man wichtige Daten, auch im Falle eines atomaren Angriffs, schützen könnte. Die aus diesen Überlegungen resultierende Grundidee war, die Daten redundant, auf mehreren Rechnern gleichzeitig zu halten. Um die Aktualität der Daten gewährleisten zu können, mussten diese in der Lage sein, sich selbständig, auf direktem Wege, abzugleichen. Die Schlussfolgerung aus diesen Anforderungen war, dass man ein Netzwerk benötigte, das diese Rechner verbinden konnte. Eine weitere wichtige Forderung war, dass die Daten auf mehreren unterschiedlichen Wegen, von einem Rechner zum anderen gelangen konnten, damit auch im Falle eines Ausfalls einzelner Rechner, die Übertragung noch möglich war. Somit musste etwas bisher noch nicht dagewesenes geschaffen werden. Das ARPANet stellt den Ursprung des heutigen Internets dar. Es wurde in den 1960er Jahren, während des kalten Kriegs, zwischen den USA und der UdSSR, von den Amerikanern, entwickelt. Seinen Namen verdankt es der ARPA (Advanced Research Projects Agency), einer Gruppe von Wissenschaftlern, die 1962, unter der Leitung des Massachusetts Institute of Technology und des US-Verteidigungsministeriums, für die US-Luftwaffe, dieses Netzwerk entwickelten. Entscheidend für seine Entstehung waren die beiden Forscher Paul Baran und Donald Watts Davies. Sie steuerten wichtige Impulse zur Entwicklung bei.

Wie ging es weiter?

Bald sollte sich herausstellen, dass ein solches Netzwerk nicht nur für militärische Zwecke interessant war, sondern auch für die zivile Wissenschaft. Wissenschaftler in den USA hatten in den 1970er Jahren die Möglichkeit, Daten mit anderen Instituten, über das ARPANet, auszutauschen. Nicht die Tatsache, Daten redundant auf mehreren Rechnern halten zu können, sondern der Datenaustausch an sich waren für sie interessant. Zu Beginn der 1980er Jahre erkannte man, dass durch die nun deutlich gestiegene Rechnerzahl im ARPANet, und die vermischte zivile und militärische Nutzung, eine Aufteilung des Netzwerks erforderlich wurde. Nur so war es den Militärs möglich Geheimnisse, und eigene Interessen zu wahren. Für den militärischen Datenaustausch wurde das "MilNet" geschaffen, das ARPANet wurde vollständig der zivilen Nutzung überlassen. Durch diese Abkopplung stand einem weiteren Anwachsen des

ARPANets nichts mehr im Wege. Und so bürgerte sich nach und nach der Name "Internet" für dieses Netzwerk ein. Der Name ARPANet verschwand im Laufe der 80er Jahre. Eng verknüpft mit dem Siegeszug des Internets, war die Entstehung des Datenaustauschprotokolls, TCP/IP. Durch diese Abkopplung stand einem weiteren Anwachsen des ARPANets nichts mehr im Wege. Und so bürgerte sich nach und nach der Name "Internet" für dieses Netzwerk ein. Der Name ARPANet verschwand im Laufe der 80er Jahre. Eng verknüpft mit dem Siegeszug des Internets, war die Entstehung des Datenaustauschprotokolls, TCP/IP.

Weitere Informationen zur Entstehung des Internet

Informationen zu TCP/IP und Weiteres zur Entstehung des Internets, sind in der Tour 2 des Einstiegskapitels von SELFHTML zusammengefasst.

Erarbeiten Sie sich die in Tour 2 bei SELFHTML aufbereiteten Informationen zum Thema Grundlagen des Internets und betrachten Sie sie als Hintergrundwissen bzw. Basis der folgenden Kapitel. [17, SELFHTML - Tour 2]

1.1 Testseite

1.2 Testseite

Listing 1.1

```
1 | <div class="blackbox"></div>
```

HTML

2 HTML

2.1 HTM...Was?



2.2 Tags

geSHi HTML TEST

ein div

Listing 2.1

```
1 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
2 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
3 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
4 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
5 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
6 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
7 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
8 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
9 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
10 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
11 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
12 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
13 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
14 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
15 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
16 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
17 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
18 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
19 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
20 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
21 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
22 <div class="blackbox" style="box-shadow:2px 2px 2px #333"></div>
```

HTML

geSHi CSS TEST

Listing 2.2

```

1  table.html5 {
2      &:after {
3          position: absolute;
4          top: 20px;
5          right: -45px;
6          width: 150px;
7          height: 25px;
8          line-height: 25px;
9          font-size: 1.2em;
10         font-weight: bold;
11         color: #fff;
12
13         //    CHANGE TO RIGHT LANGUAGE    \\
14         content: "HTML";
15
16
17
18         @include text-shadow(#777 1px 1px 1px);
19         background-color: rgb(255,183,107);
20         @include filter-gradient(#ffb76b, #ff7f04, vertical);
21         text-align: center;
22         @include box-shadow(0px 1px 5px #555);
23         -webkit-transform: rotate(45deg);
24         -moz-transform: rotate(45deg);
25         -o-transform: rotate(45deg);
26         -ms-transform: rotate(45deg);
27         transform: rotate(45deg);
28     }
29 }

```

CSS

geSHi PHP TEST

Listing 2.3

```

1  function sys_securePage ()
2  {
3      echo "Content of the string";
4      $value = is_file($value);
5  }

```

PHP

2.3 Das Grundgeruest

2.3.1 HEAD und BODY

Kopf und Körper - HEAD und BODY


Das Grundgerüst einer HTML-Datei setzt sich aus drei Teilen zusammen:

- der Deklaration des Dokumenttyps,
- dem Kopf der HTML-Datei (Header)
- und dem Körper der HTML-Datei, dem Body.

Jeder dieser drei Teile wird in Form eines Tags angegeben bzw. von einem öffnenden und einem schließenden Tag umgeben.

Listing 2.4

```
1  <!doctype html>
2  <html>
3
4      <head>
5          <title>Der Titel der HTML - Seite </title>
6      </head>
7
8      <body>
9
10         </body>
11
12 </html>
```



2.3.2 Absaetze

2.4 DOCTYPE

2.5 Sonderzeichen

2.6 Zeilenumbrueche

Grundsätzlich wird ein Text immer am Ende des Browserfensters bzw. am Ende des umgebenden Blocks (z. B. der Tabellenzelle) automatisch an einem Leerzeichen umgebrochen. Um einen Zeilenumbruch an einer bestimmten Stelle zu erzwingen, verwendet man das Tag BR

Listing 2.5

HTML

```
1 <h1>Ohne UmbrÄ¼che</h1 >
2 <p>Dieser Text steht in einem ersten Absatz .</p>
3 <p>Dieser Text in einem weiteren .</p>
4 <p>In diesem dritten Absatz ist nun noch etwas mehr Text vorhanden .</p>
5 <h1>Mit UmbrÄ¼mchen </h1 >
6 <p>Dieser Text steht in <br > einem ersten Absatz .</p>
7 <p>Dieser Text <br > in einem weiteren .</p>
8 <p>In diesem dritten <br > Absatz ist nun noch etwas mehr Text
9 vorhanden .</p>
```

3 CSS

jeweils viele kleine Nachfrager

Zahl der Anbieter	vollkommener Markt	unvollkommener Markt
viele kleine	vollständige Konkurrenz	monopolistische Konkurrenz
wenige mittlere	homogenes Oligopol	heterogenes Oligopol
ein großer	Monopol	monopolistische Preisdifferenzierung

egalitärer Liberalismus vs. Naturrechtslibertarismus

	Liberalismus	Libertarismus
Privateigentum	pragmatisch (Maximin)	unverletzlich
öffentliche Produktion	pragmatisch (Maximin)	abgelehnt
Steuern	probates politisches Mittel	staatlicher Diebstahl
Privateigentum	pragmatisch (Maximin)	unverletzlich
Umverteilung	Befürwortung (Maximin)	Verteilungsgerechtigkeit irrelevant
Soziale Sicherungssysteme	unterstützt	abgelehnt

4 WEBSERVER

4.1 HTTP und das Web

4.1.1 Webserver und Protokolle

Was ist ein Webserver?

Betrachtet man nur die physikalischen Komponenten des Internets, dann ist dieses Netzwerk eine gigantische Ansammlung von aktiven Komponenten (Switches, Router, usw.) und Computern, die verschiedenste Dienste in diesem Netzwerk zur Verfügung stellen (Webserver). Was aber ist ein Webserver genau?

- Was ist die Hauptaufgabe eines Webserver?
- Nennen Sie drei Protokolle, die häufig zur Kommunikation mit einem Webserver benutzt werden!
- Auf welcher Schicht im ISO/OSI-Modell ist ein Webserver einzuordnen? Begründen Sie Ihre Antwort!
- Wo, außer im Internet, kann man Webserver finden? Nennen Sie mindestens drei Möglichkeiten!

Um Nachrichten, Webseiten oder Binärdateien mit anderen Webservern oder Browsern austauschen zu können, benötigen Webserver Protokolle, wie z. B. das "Hyper Text Transfer Protocol", kurz HTTP.

- Was ist ein Protokoll (Artikel "Kommunikationsprotokoll")?
- Welche Hauptaufgabe hat das Protokoll HTTP (Artikel "HTTP")?
- Zu welcher Protokollfamilie gehört HTTP?
- Auf welchen Schichten des ISO/OSI-Modells ist HTTP beheimatet?
- Wie werden die Kommunikationseinheiten in HTTP bezeichnet?
- Wie viele unterschiedliche Arten solcher Kommunikationseinheiten gibt es in HTTP? Nennen Sie auch deren Namen!
- Was ist ein "HTTP-Header", und was ist ein "Message-Body"? Welche Informationen enthalten HTTP-Header bzw. Message-Body?
- Was bedeuten die Angaben "HTTP-Get" und "HTTP-Post"? Worin unterscheiden sich diese beiden Dinge?
- Worin liegt der Unterschied zwischen den Versionen "HTTP 1.0" und "HTTP 1.1"

- Wie heißen die Teile, in die sich der obige URI gliedert (Hinweis: Es sind fünf Teile)?
- Worin liegt der Unterschied zwischen einem URI und einem URL?

Es gibt verschiedene Versionen des Apache. Am häufigsten werden die Versionen 1.3, 2.0 sowie 2.2 genutzt, wobei zwischen den Versionen 1.3 und 2.0 ein großer Qualitätssprung stattfand. Bei einer Neuinstallation sollte darauf geachtet werden, immer die aktuellste Version, mit ihren aktuellsten Patches, zu nutzen, damit Sicherheitslücken so schnell wie möglich geschlossen werden. Die derzeit aktuellste Version des Apache ist 2.4.2.

Recherchieren Sie im Internet, wo Sie...

- die aktuelle Online-Dokumentation zum Apache HTTP-Server finden.
- das aktuelle Installationspakete für Windows (inklusive OpenSSL) finden.

Speichern Sie sich diese Links als Bookmarks auf Ihrem PC!

Installation

Die Echtheit des Installationspaketes überprüfen

Laden Sie die aktuellste Version (2.4.x) des Apache HTTP-Servers für Windows, und die dazugehörige SHA1-Datei herunter! (Download here)

Mit Hilfe der SHA1-Checksumme einer Datei kann deren Echtheit verifiziert werden. Für das Installationspaket wird die Checksumme "64D78A9C90E005E8F4F55F4E1C3720E856BBC005" bereitgestellt. Um eine Überprüfung dieses Pakets durchführen zu können, wird zusätzlich noch eine Software benötigt, die SHA1-Checksummen generieren kann. Eine solche Software ist z. B. "digestIT2004".

Laden Sie die Software "digestIT2004" aus dem Internet herunter (Download here), und installieren Sie sie (Complete Installation). Führen Sie jetzt eine Verifikation des Apache-Installationspaketes, gemäß der folgenden Anleitung durch!

1. Markieren und Kopieren Sie sich die SHA1-Checksumme (64D78A9C90E005E8F4F55F4E1C3720E856BBC005).
2. Klicken Sie mit der rechten Maustaste auf das Installationspaket. Es erscheint ein Kontextmenü.
3. Klicken Sie auf den Menüeintrag "digestIT 2004" mit der linken Maustaste.
4. Wählen Sie den Menüpunkt "Verify SHA1 Hash"!

Wenn alles ordnungsgemäß verlaufen ist, sollte die Meldung "Digest matches. Verification succeeded." erscheinen, und die Verifikation ist erfolgreich abgeschlossen.

Die Software installieren

Installieren Sie, nach den folgenden Vorgaben, den Apache HTTP-Server auf Ihrem Laptop!

- Network Domain: apache.org
- Server Name: localhost
- Administrator's Email Address: Admin@localhost.apache.org
- Install for all users, on port 80, as a service -- Recommended.
- Setup Type: Typical
- Installationspfad: DATENPARTITION\apache\httpd\2.4.2

Ist die Installation erfolgreich verlaufen, erscheint am unteren rechten Bildschirmrand ein kleines Symbol, eine Feder mit einem grünem Pfeil.

Die Steuerkonsole (Apache Service Monitor)

Mit einem Doppelklick können Sie den Apache Service Monitor öffnen.

Welche Möglichkeiten/Funktionen bietet der Apache Service Monitor?

Der Apache in der Windows Dienstkonsole

Wenn der Apache HTTP-Server als Dienst installiert wird, ist er selbstverständlich auch in der Windows Dienstkonsole zu finden.

Unter welchem Namen ist der HTTP-Server in der Windows Dienstkonsole zu finden?

Nach einem Doppelklick auf den HTTP-Server Dienst öffnet sich dessen Eigenschaftenfenster. Achten Sie an dieser Stelle darauf, dass der Apache automatisch, beim Systemstart, mitgestartet wird.

Das Installationsverzeichnis

Nach der Installation des HTTP-Servers finden sich in dessen Installationsverzeichnis diverse Unterverzeichnisse. Diese sind:

Welche Bedeutung haben die unten aufgeführten Unterverzeichnisse?

bin

cgi-bin

conf

error

logs

modules

htdocs

4.2.2 Konfiguration

Der erste Blick in die Konfigurationsdatei

Der Apache HTTP-Server wird nicht wie viele andere Windowsprogramme mittels der Windows Registry, sondern mit Hilfe einer eigenen Textdatei konfiguriert. Üblicher weise heißt diese Datei httpd.conf. Sie kann jedoch auch einen beliebigen anderen Namen tragen und durch

zusätzliche Dateien erweitert werden. Zu finden ist diese Datei im Unterverzeichnis "conf" des Installationsverzeichnis.

Fertigen Sie jetzt eine Sicherheitskopie der Datei httpd.conf an!

Direktiven

Zur Konfiguration des Apache werden sogenannte "Direktiven" in die Datei httpd.conf eingetragen. Eine solche Direktive ist wie folgt aufgebaut:

```
Direktive  Argument
ServerRoot /usr/local/apache2
```

Für die Erstellung von Direktiven gelten die folgenden Regeln:

- Nur eine Direktive pro Zeile
- Die Direktiven sind nicht Casesensitiv, d. h. ServerRoot = serverroot.
- Die Argumente der Direktiven sind Casesensitiv.
- Einrückungen können verwendet werden, um die Lesbarkeit der Konfigurationsdatei zu verbessern.
- Der Backslash (\) kann verwendet werden, um sehr lange Zeilen in mehrere Zeilen zu verteilen (Linebreak char).
- Bei der Angabe von Datei- und Verzeichnispfaden wird immer der Slash (/) als Trennzeichen verwendet, nicht der Backslash (\).

Kommentare

Recherchieren Sie in der Onlinedokumentation des Apache HTTP-Server, wie Kommentare in die Konfigurationsdatei eingetragen werden, bzw. welche Regeln dafür gelten!

Erstellen einer ersten Konfiguration

Noch startet der Server nicht

Die Mindestkonfiguration eines Apache HTTP-Servers besteht aus der Listen-Direktive.

- Finden Sie heraus, welche Bedeutung und welche Syntax die Listen-Direktive hat!
- Legen Sie eine neue Konfigurationsdatei, unter dem Namen httpd.conf an! Der HTTP-Server soll ausschließlich auf Port 80 lauschen!
- Testen Sie, ob Ihr Apache HTTP-Server startet!
- Welche Fehlermeldung erhalten Sie von Ihrem Browser, wenn Sie die URL http://localhost eingeben?

Aktuell ist der Apache noch nicht in der Lage, Webseiten auszuliefern. Dies liegt daran, dass der Server keinerlei Information darüber hat, in welchem Verzeichnis die Dateien liegen, mit denen er arbeiten soll. Die betreffende Information erhält der HTTP-Server mit der DocumentRoot-Direktive. Sie nimmt eine Pfadangabe als Argument entgegen.

Standardmäßig ist das Unterverzeichnis htdocs im Apache Installationsverzeichnis als DocumentRoot vorgesehen.

- Fügen Sie die DocumentRoot-Direktive in Ihre Konfigurationsdatei ein, und verweisen Sie mit ihr auf das htdocs-Verzeichnis (z. B. E:/apache/httpd/2.4.2/htdocs)!
- Testen Sie erneut den Zugriff auf <http://localhost> mit Hilfe Ihres Browsers! Welche Antwort erhalten Sie jetzt?
- Versuchen Sie jetzt, mit Hilfe des Internet Explorers, auf die Adresse <http://localhost/index.html> zuzugreifen! Welche Antwort erhalten Sie von Ihrem Browser?
- Versuchen Sie erneut auf die Adresse <http://localhost/index.html> zuzugreifen, dieses mal aber mit Hilfe des FireFox! Wie unterscheiden sich die Antworten des FireFox und des Internet Explorers von einander?

Weitere wichtige Direktiven

Mit den beiden Direktiven Listen und DocumentRoot hat der Apache HTTP-Server bereits alle Informationen, die er für einen rudimentären Betrieb benötigt. Für den produktiven Betrieb eines Apache HTTP-Servers genügen diese Angaben jedoch noch nicht. Es existieren weitere Direktiven, die zur Mindestkonfiguration dieses Webservers gehören. Die folgende Auflistung zeigt einige davon.

Recherchieren Sie die Bedeutung der Direktiven in der obigen Auflistung!

Direktive	Bedeutung
DefaultType	
ErrorLog	
LogLevel	
ServerAdmin	
ServerName	
ServerRoot	

Einbinden von Modulen

Wie bereits zu Anfang erwähnt, ist der Apache HTTP-Server ein modular aufgebautes Produkt. Im Unterverzeichnis `modules`, des Installationsverzeichnis, liegen mehrere Dateien, mit der Dateierweiterung `.so`. Hierbei handelt es sich um sogenannte "Shared Objects", oder auch um "Dynamic Link Libraries", die in der Programmiersprache C erstellt wurden. Jede dieser Dateien stellt eine Vielzahl an Funktionen bereit, die der Apache nutzen kann, um seine eigenen Fähigkeiten zu erweitern.

- Recherchieren Sie in der Onlinedokumentation (Dynamic Shared Objects), wie das Laden von Modulen funktioniert (Syntax)!

- Recherchieren Sie in der Dokumentation was die Direktive `DirectoryIndex` `index.html` bedeutet!
- Welches Modul muss geladen werden, damit die `DirectoryIndex`-Direktive funktioniert?
- Laden Sie das betreffende Modul, tragen Sie diese Direktive in Ihre Datei `httpd.conf` ein und prüfen Sie was passiert, wenn Sie die URL `http://localhost`, nach einem Webserverneustart, aufrufen!

Bedingtes laden von Modulen

In manchen Umgebungen kann es notwendig sein, dass ein Apache-Modul nur dann geladen werden darf, wenn eine bestimmte Bedingung erfüllt ist. Um dies zu erreichen, muss der `IfModule`-Kontainer eingesetzt werden.

Ein Kontainer besteht aus einem öffnenden Tag (z. B. `IfModule [Bedingung]`) und einem schließenden Tag (`/IfModule`). Beide Tags umschließen Direktiven, die nur dann ausgeführt werden, wenn der betreffende Kontainer bei einer Anfrage angesprochen wird.

Im konkreten Falle des `IfModule`-Kontainers heißt das, dass die in einem solchen Kontainer eingeschlossenen Direktiven nur dann ausgeführt werden, wenn `[Bedingung]` zutrifft. Das Beispiel auf dieser Seite zeigt den `IfModule`-Kontainer, zusammen mit der `DirectoryIndex`-Direktive. Diese wird nur dann ausgeführt, wenn das Modul `dir_module` geladen wurde.

Listing 4.1

```
1 <IfModule dir_module>
2     DirectoryIndex index.html
3 </IfModule>
```

4.2.3 Sicherheit

Directory-Kontainer

Directory-Kontainer stellen einen zentralen Sicherheitsmechanismus innerhalb der Konfiguration des Apache HTTP-Servers dar. Mit ihrer Hilfe kann der Zugriff auf Verzeichnisse, die über den Webserver erfolgen, reguliert werden. Wie bereits beim `IfModule`-Kontainer besprochen, gelten auch hier die Anweisungen innerhalb des Kontainers nur für das angegebene Verzeichnis.

- Erläutern Sie die Bedeutung der Anweisungen im Directory-Kontainer aus dem unten angegebenen Beispiel.
- Welches Modul muss geladen sein, damit die Direktive `Require` zur Verfügung steht?
- Laden Sie `SELFHTML` aus dem Internet herunter, falls dies noch nicht geschehen ist, und entpacken Sie das ZIP-File in das Verzeichnis `htdocs/selfhtml`.
- Erstellen Sie in der Datei `httpd.conf` einen Directory-Kontainer, der den Zugriff auf das Verzeichnis `htdocs/selfhtml` von allen jedem Rechner aus und ohne jede weitere Bedingung erlaubt.

Listing 4.2

```
1 <Directory />
2     AllowOverride none
3     Require all denied
4 </Directory>
5
```

Externe Konfiguration der Verzeichnissicherheit

In manchen Situationen kann es notwendig sein, dass die Konfiguration der Zugriffsrechte eines Verzeichnisses nicht durch den Administrator des Webserver, sondern durch eine andere Person erfolgen muss. In so einem Fall gibt es zwei Optionen:

- Der Serveradministrator gibt der betreffenden Person Zugriff auf die Hauptkonfiguration des Webserver (httpd.conf).
- Es werden .htaccess-Dateien genutzt (man beachte den Punkt vor dem Namen).

Im Allgemeinen dürfte die zweite Option, die Nutzung der .htaccess-Dateien, die sinnvollere Alternative sein.

Unter MS-Windows ist es nicht möglich, dass ein Dateiname mit einem Punkt beginnt. Deshalb muss hier die AccessFile-Direktive in der Hauptkonfigurationsdatei genutzt werden, um den Namen der .htaccess-Dateien zu ändern, z. B. in htaccess (ohne Punkt)!

Benutzung von .htaccess-Dateien

.htaccess-Dateien ermöglichen es, den Zugriff auf die Verzeichnisstruktur des Webserver auf der Basis einzelner Verzeichnisse zu regeln. Dies geschieht, indem in den .htaccess-Dateien in die Verzeichnisse gelegt werden, für die Zugriffsbeschränkungen eingerichtet werden müssen.

Die Direktiven einer .htaccess-Datei gelten immer für das Verzeichnis, in dem sie sich befindet, und alle darunter liegenden Unterverzeichnisse.

.htaccess-Dateien nutzen die gleiche Syntax, wie die Hauptkonfigurationsdatei httpd.conf. So kann z. B. der Directory-Kontainer für das Verzeichnis htdocs/selfhtml durch eine entsprechende .htaccess-Datei ersetzt werden.

Innerhalb der .htaccess-Datei entfällt die Angabe des Directory-Kontainers, da der gesamte Inhalt der Datei nur für ein Verzeichnis gültig ist.

Testen Sie, ob die auf dieser Seite gezeigte .htaccess-Datei bereits Auswirkungen hat! Wenn nicht, was muss an der Hauptkonfigurationsdatei geändert werden, um die .htaccess-Datei wirksam werden zu lassen?

Negative Effekte

Bei der Nutzung von .htaccess-Dateien treten im Wesentlichen drei negative Effekte auf:

1. Sobald mit Hilfe der AllowOverride-Direktive die Nutzung von .htaccess-Dateien erlaubt wird, sucht der Apache automatisch, in allen Unterverzeichnissen immer nach einer .htaccess-Datei, auch dann, wenn gar keine vorhanden ist.
2. Eine .htaccess-Datei wird bei jedem Verzeichnisszugriff erneut geladen, um etwaige Änderungen an der Verzeichniskonfiguration feststellen zu können.
3. Da die Direktiven aus den .htaccess-Dateien immer kumulativ betrachtet werden, müssen alle .htaccess-Dateien, auch die aus höheren Ebenen gelesen werden, um die Direktiven addieren zu können, und somit die effektive Konfiguration bilden zu können.

Direktiven aus verschiedenen .htaccess-Dateien werden immer in der Reihenfolge angewandt, in der sie vom Webserver vorgefunden werden. D. h. Direktiven aus .htaccess-Dateien in den

unteren Verzeichnisebenen können Direktiven aus .htaccess-Dateien den höheren Verzeichnisebenen überschreiben.

Ein Beispiel zum vorangegangenen Merksatz:

Es existieren die folgenden Verzeichnisse: `htdocs/selfhtml` und `htdocs/selfhtml/css`. Beide Verzeichnisse haben eine eigene .htaccess-Datei. In der .htaccess-Datei des Verzeichnisses `htdocs/selfhtml` ist die Direktive `Require all granted` angegeben. Das bedeutet, dass ein Nutzer Zugriff auf beide Verzeichnisse hat, da die Direktiven aus .htaccess-Dateien auch immer für alle Unterverzeichnisse gelten.

Im Verzeichniss `htdocs/selfhtml/css` ist jedoch in der .htaccess-Datei die Direktive `Require all denied` gesetzt. Daraus folgt, dass der Zugriff auf das Verzeichnis `htdocs/selfhtml/css` für alle Nutzer gesperrt ist, weil die `Require all denied`-Direktive als zweites gefunden wird, und somit Gültigkeit hat. Nutzer können also nur auf das Verzeichnis `htdocs/selfhtml` zugreifen.

1. Wie verhält es sich zwischen einer .htaccess-Datei und der Hauptkonfigurationsdatei? Kann eine .htaccess-Datei Direktiven aus der Hauptkonfigurationsdatei überschreiben, und umgekehrt?
2. Was passiert, wenn ein Benutzer, in seinem Browser, die folgende URL aufruft: `http://localhost/selfhtml/htaccess` oder `http://localhost/selfhtml/.htaccess`?
3. Wie kann verhindert werden, dass sich ein Nutzer den Inhalt einer .htaccess-Datei anzeigen lässt (Hinweis: Schlagen Sie hierfür die `FilesMatch`-Direktive nach!)

Listing 4.3

```

1 # httpd.conf
2 <Directory "htdocs/selfhtml">
3     Require all granted
4 </Directory>
5
6 #.htaccess-Datei
7     Require all granted

```

HTML

4.2.4 Indexes

Ein einfacher Verzeichnisindex

Webseiten darzustellen ist nur eine Aufgaben, von vielen, die ein Webserver zu bewältigen hat. Eine andere, ist das bereitstellen von Dateien zum Download.

1. Erstellen Sie den Ordner `htdocs/download`.
2. Verschieben Sie die `SELFHTML`-Zip-Datei in den Downloadordner.
3. Konfigurieren Sie den Downloadordner so, dass alle Nutzer darauf zugreifen können (Zugriffsrechte haben)!
4. Laden Sie das Module "`mod_autoindex.so`"! Welche Aufgabe hat dieses Modul?
5. Testen Sie den Zugriff auf die URL: `http://localhost/download`!
6. Falls der Zugriff noch nicht funktionieren sollte: Welche zusätzliche Direktive müssen Sie angeben, damit eine automatische Indizierung des Downloadordners funktioniert?

Benutzen Sie jetzt eine .htaccess-Datei, um den Zugriff auf den Downloadordner zu ermöglichen! Welche Änderungen müssen dazu an der Hauptkonfigurationsdatei gemacht werden?

Ein bisschen Farbe ins Spiel bringen

Das Modul "mod_autoindex.so" bietet viele verschiedene Möglichkeiten, um einen einfachen, schmucklosen Verzeichnisindex ansprechender zu gestalten. Eine dieser Möglichkeiten ist beispielsweise das hinzufügen von Icons. Hierzu muss jedoch zu erst die Option FancyIndexing aktiviert werden.

Aktivieren Sie die Index-Options FancyIndexing! Was verändert sich dadurch sofort am Index?

Listing 4.4

```

1 <Directory "htdocs/download">
2     Require all granted
3
4     # FancyIndexing aktivieren
5     IndexOptions FancyIndexing
6
7     Options          Indexes
8 </Directory>
9
```

PHP

Icons hinzufügen

Der Apache HTTP-Server kennt zwei Wege, um Dateitypen mit Icons zu verbinden:

- Durch Angabe des Dateityps
- Durch Angabe eines MIME-Typs

Hierfür existieren drei Direktiven:

Direktive	Bedeutung
AddIcon	Verbindet verschiedene Dateitypen mit einem Icon. AddIcon nimmt den Namen einer Icon-Datei und eine oder mehrere Dateieindungen entgegen.
AddIconByEncoding	Verknüpft Dateien mit Hilfe des MIME-Typs mit einem Icon. Hier wird der Name einer Icon-Datei und ein MIME-Type entgegen genommen.
AddIconByType	Verhält sich genau so, wie AddIconByEncoding

Was ist ein MIME-Type?

Da mit der HTTP-Server MIME-Typen verarbeiten kann, muss das Modul "mod_mime.so" geladen werden. Dieses Modul, wird zusammen mit einer Konfigurationsdatei für MIME-Typen geliefert. Um diese Datei zu laden, müssen die folgenden Direktiven in die Hauptkonfigurationsdatei aufgenommen werden:

Nehmen Sie die Direktiven aus \beispiel{TypesConfig} in Ihre Hauptkonfigurationsdatei auf, und benutzen Sie die Direktive `AddIconByType`, um das Icon "compressed.gif" für ZIP-Dateien festzulegen! Welchen MIME-Type benötigen Sie hierfür? Testen Sie Ihr Ergebnis!

Listing 4.5

```
1 LoadModule mime_module "modules/mod_mime.so"
2 TypesConfig "conf/mime.types"
```

PHP

Aliase / Virtuelle Verzeichnisse

Das das Laden des Icons in der letzten Übung fehlschlug lag daran, dass die benötigte Ressource, das Icon, außerhalb des DocumentRoot-Verzeichnisses liegt. Um Ressourcen außerhalb des DocumentRoot erreichen zu können, muss mit Aliasnamen bzw. Virtuellen Verzeichnissen gearbeitet werden.

Verzeichnisalias ermöglichen es, Ressourcen außerhalb des DocumentRoot-Verzeichnisses einzubinden. Zuständig ist hierfür das "alias_module" (mod_alias.so).

Laden Sie das Modul mod_alias.so, und recherchieren Sie, welche Direktive für das Erstellen eines Verzeichnisaliases zuständig ist.

Prüfen Sie, ob im Verzeichnisindex jetzt ein Icon angezeigt wird! Falls nicht, kontrollieren Sie Ihr Error-Log!

Um den Zugriff auf die Icon-Ressource zu ermöglichen, muss dem Nutzer der Zugriff auf das icons-Verzeichnis gewährt werden. Dies geschieht wiederum mit einem Directory-Kontainer.

Worin unterscheiden sich die beiden folgenden Directory-Kontainer (Hinweis: Benutzen Sie Ihr Error-Log, um den Unterschied festzustellen)?

Formatieren Sie den automatischen DirectoryIndex des Downloadordners nach den folgenden Angaben. Benutzen Sie die Onlinedokumentation zum "autoindex_module", um die benötigten Informationen zu recherchieren!

- Als Icon für das übergeordnete Verzeichnis soll dir.gif verwendet werden!
- Für *.msi-Dateien soll das Icon compressed.gif genutzt werden.
- Das Standardicon für alle unbekannten Dateitypen muss unknown.gif sein.
- Im Index sollen Verzeichnisse immer an erster Stelle aufgelistet werden, und dann erst die Dateien.
- Die Icons müssen ein Teil der Links sein.
- Dateinamen müssen in ihrer vollen Länge erhalten bleiben und dürfen nicht abgeschnitten werden.
- Die Spalte "Description" soll ausgeblendet werden.
- Die Standardsortierung für den Verzeichnisindex ist eine aufsteigende Reihenfolge.

Listing 4.6

```
1 <Directory "/icons">
2     Require all granted
3 </Directory>
4
5 <Directory "E:/apache/httpd/2.4.2/icons">
6     Require all granted
7 </Directory>
8
```

PHP

4.2.5 SSI

4.2.6 CGI