



# Webdesign

---

FüUstgSBw  
First Edition

HF Weidinger  
L Siegerth

---

# Inhaltsverzeichnis

---

|  |    |
|--|----|
| 1 Grundlagen                             | 4  |
| 1.1 Coding Guidelines                    | 4  |
| 1.1.1 HTML                               | 4  |
| 1.1.2 CSS                                | 4  |
| 1.2 Entstehung des Internet              | 4  |
| 2 HTML                                   | 5  |
| 2.1 Grundlagen HTML                      | 6  |
| 2.2 Inhalte in HTML                      | 9  |
| 2.2.1 Zeilenumbrueche                    | 9  |
| 2.2.2 Tables                             | 10 |
| 2.2.3 Headings                           | 15 |
| 2.2.4 Paragraphs                         | 16 |
| 2.2.5 Images                             | 18 |
| 2.2.6 Links                              | 18 |
| 2.2.7 Formulare                          | 20 |
| 2.2.8 Kommentare                         | 22 |
| 2.2.9 Textformatierung                   | 22 |
| 2.2.10 Listen                            | 23 |
| 2.3 Struktur in HTML                     | 24 |
| 2.3.11 BLOCK                             | 24 |
| 2.3.12 Blockelemente vs. Inline Elemente | 24 |
| 2.3.13 Semantische Strukturierung        | 26 |
| 3 CSS                                    | 28 |
| 3.1 Schriften                            | 28 |

|   |    |
|---|----|
| 3.2 CSS3 - Effekte . . . . .                  | 28 |
| 3.2.14 text-shadow . . . . .                  | 28 |
| 3.2.15 box-shadow . . . . .                   | 31 |
| 3.2.16 border-radius . . . . .                | 31 |
| 4 WEBSERVER . . . . .                         | 32 |
| 4.1 HTTP und das Web . . . . .                | 32 |
| 4.1.17 Webserver und Protokolle . . . . .     | 32 |
| 4.2 Apache . . . . .                          | 35 |
| 4.2.18 Virtual Hosting . . . . .              | 35 |
| 4.2.19 Einfuehrung und Installation . . . . . | 38 |
| 4.2.20 Konfiguration . . . . .                | 42 |
| 4.2.21 Sicherheit . . . . .                   | 46 |
| 4.2.22 Indexes . . . . .                      | 52 |
| 4.2.23 SSI . . . . .                          | 60 |
| 4.2.24 CGI . . . . .                          | 63 |
| 4.2.25 PERL mit mod_perl.so . . . . .         | 67 |
| 4.2.26 Apache und PHP . . . . .               | 70 |
| 4.2.27 Die Log-Dateien des Apache . . . . .   | 72 |

# 1 Grundlagen

---

## 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 Coding Guidelines

### 1.1.1 HTML

### 1.1.2 CSS

## 1.2 Entstehung des Internet

## 2 HTML

### Was ist HTML

HTML ist die Auszeichnungssprache für die Inhalte einer Webseite. Mittels HTML wird die Strukturierung von Inhalten wie Texten, Bildern, Hyperlinks oder anderen Ressourcen vorgenommen.

HTML-Dokumente .html .htm bilden die Grundlage des World Wide Web WWW. Es ist möglich in HTML nicht nur die sichtbaren Inhalte zu strukturieren, sondern auch zusätzliche Informationen, wie zum Beispiel den Autor, das Datum oder eine Beschreibung der Seite zu hinterlegen.



Die Definitionen, wie die Sprache aufgebaut ist und auch die Weiterentwicklung unterliegt dem World Wide Web Consortium W3C.

Derzeit ist die aktuelle Version HTML5

Das Acronym HTML steht für HyperText Markup Language

### Websites, Homepages, Webpages, etc....

Die oft gehörten Begriffe

- Website,
- Homepage,
- Webpage,
- Internetseite,
- HTML-Seite und
- Webauftritt



stellen im groben Sinne Synonyme dar.

Im normalen Sprachgebrauch kann und wird zwischen diesen Begriffen nicht unterschieden. Im engeren Sinne lassen sich jedoch kleinere Unterschiede feststellen. Beispiel: Ein **Webauftritt** kann aus mehreren einzelnen HTML-Seiten bestehen. Beispiel 2: Korrekt betrachtet ist eine **Homepage** nur eine einzelne Seite. Erst wenn es mehrere Pages sind, kann von einer Site gesprochen werden.

Da umgangssprachlich diese Abgrenzungen jedoch nicht berücksichtigt werden, ist dies relativ irrelevant

## 2.1 Grundlagen HTML

### Versionen von HTML

Die Standards für HTML werden durch das World Wide Web Consortium W3C entwickelt und veröffentlicht.



| Version   | Erscheinungsdatum | Merkmale  |
|-----------|-------------------|---|
| HTML      | 03.11.1992        | Initial Release                                       |
| HTML2     | 11.1995           | Formulare werden eingeführt                           |
| HTML3.2   | 01.1997           | Tabellen, Textfluss um Bilder, Einbindung von Applets |
| HTML4     | 12.1997           | Stylesheets, Skripte und Frames                       |
| XHTML 1.1 | 2001              | Eine Neuformulierung von HTML 4.01 mit Hilfe von XML  |
| HTML 5    | 04.2009           |   |

### HTML - Struktur

Eine HTML Datei besteht grundsätzlich aus 3 Bereichen:

1. der Dokumenttypdeklaration (**Doctype**) ganz am Anfang der Datei, die die verwendete Dokumenttypdefinition DTD angibt, z. B. **HTML 4.01 Strict**,
2. dem HTML-Kopf (**HEAD**), der hauptsächlich technische oder dokumentarische Informationen enthält, die üblicherweise nicht im Anzeigebereich des Browsers dargestellt werden
3. dem HTML-Körper (**BODY**), der jene Informationen enthält, die gewöhnlicherweise im Anzeigebereich des Browsers zu sehen sind.

Damit gelangen wir zu folgender Struktur:

Listing 2.1

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2  <html>
3
4      <head>
5      </head>
6
7      <body>
8      </body>
9
10 </html>

```

### HTML - HEAD

Der head - Bereich einer HTML-Datei dient der Übergabe von zusätzlichen Daten, die nicht direkt zum anzuzeigenden Inhalt gehören. Diese Informationen können zum Beispiel externe Ressourcen wie CSS - oder Javascript-Dateien sein. Oder es können Informationen über den Author, das Erstellungsdatum, oder eine Beschreibung sein.

Mögliche Elemente im head

| Element | Bedeutung | Anwendung   |
|---------|-----------|---|
| title   | TITEL     | Der in den Tabs eines Browsers angezeigte Titel der Seite |

| Element | Bedeutung                             | Anwendung  |
|---------|---------------------------------------|--|
| meta    | ZUSATZINFORMATIONEN                   | z.B. über den Author der Seite oder das Veröffentlichungsdatum |
| base    | BASIS-URI oder -Frame                 |  |
| link    | Verknüpfung zu einer Ressource        | z.B. externe Stylesheets                                       |
| script  | CODE einer anderen Programmiersprache | hauptsächlich für die Einbindung von Javascript genutzt        |
| style   | Layout - Eigenschaften                |  |
| object  | externe Dateien                       |  |

das folgende Listing zeigt einen beispielhaften head eines üblichen HTML - Dokuments

Listing 2.2



```

1  <head>
2
3
4  <meta charset="utf-8">
5  <meta name="description" content="Development Framework">
6  <meta name='author' content='Kevin Siegerth'>
7
8
9  <title>ITSysAdminFwWebSK</title>
10
11
12 <link rel="stylesheet" media="screen" href="/css/style.css" >
13
14 <script type="text/javascript" src="etc/markItUp.js"></script>
15
16 <style type="text/css" media="print,screen">!--
17     h1 {
18         color:green;
19     }
20 --></style>
21
22 </head>

```

## HTML - BODY

Der body eines HTML - Dokuments beinhaltet die Daten, die im Browser angezeigt werden sollen.

Er beinhaltet somit alle Elemente wie:

- Links
- Tabellen
- Bilder
- Texte
- etc...

Das folgende Listing zeigt einen Beispielhaften Body für eine einfache HTML Seite.

Listing 2.3

```

1  <body>
2
3  <h1>Willkommen auf meiner Webseite</h1>
4
5  <h2>Hier habe ich eine Liste für euch erstellt</h2>
6
7  <ul>
8      <li>Links</li>
9      <li>Tabellen</li>
10     <li>Bilder</li>
11     <li>Texte</li>
12     <li>etc...</li>
13 </ul>
14
15 <h2>Und hier noch eine kleine Tabelle</h2>
16
17 <table>
18     <tr>
19         <td>Eintrag</td>
20         <td>1</td>
21     </tr>
22     <tr>
23         <td>Eintrag</td>
24         <td>2</td>
25     </tr>
26 </table>
27
28 </body>

```

HTML

## HTML - Doctype

Der DOCTYPE in HTML dient nur dem Zweck, den Rechnern, die diese Seite öffnen, eine Vorinformation zu geben. Der Computer kann sich dann direkt darauf einstellen, welche Tags folgen können und wie er diese interpretieren muss. Ein normaler Webbrowser wie Google Chrome, Firefox, Safari oder auch Opera (**NEIN**, der Internet Explorer IE ist kein **normaler** Browser) ignoriert diese Angabe gänzlich. Diese Angaben haben erst eine Relevanz, wenn automatisierte Parse-Roboter (Crawler) die Seite öffnen. Ein Crawler ist ein Programm, das eine Webseite abrufen und die ausgelesenen Inhalte verarbeitet. Im Großteil der Fälle dient der Abruf der Seiten der Indizierung der Inhalte für schnellere Suchanfragen.

Um diesen Vorgang noch zu verbessern, gibt es neue Techniken und Möglichkeiten um die semantischen Angaben einer Seite durch viele Zusatzinformationen zu optimieren. Dies wird erreicht durch zusätzlich neue Tags, die für den menschlichen Betrachter nicht sichtbar sind.

Werden zusätzliche Sprachen also in HTML eingebunden, müssen diese im DOCTYPE deklariert werden, so dass die Maschine weiß, wie Sie damit umzugehen hat.

Beispielhaft verschiedene DOCTYPEs wären:

DOCTYPE

!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd"

!doctype html

?xml version="1.0" encoding="UTF-8"?>/span>

Zelleninhalt

Zelleninhalt

Zelleninhalt



Relevant ist in diesem Lehrgang lediglich der DOCTYPE !doctype html. Dieser DOCTYPE gibt an, dass das folgende Dokument in HTML5 geschrieben wurde

## 2.2 Inhalte in HTML

### TAGS in HTML

Ein Element in HTML wird parallel zu XML meist durch einen öffnenden und schließenden Tag abgegrenzt. In HTML nennen sich diese Begrenzer **tag**. öffnender HTML Tag span schließender HTML Tag /span

daraus ergibt sich die folgende Syntax für z.B. eine Überschrift  
h1Überschrift/h1

Listing 2.4

```
1 <h1>Inhalt</h1>
2
3 <span>Inhalt</span>
4
5 <div>Inhalt</div>
```

HTML

### Selbstschließende Tags

HTML kennt zu den normalen Tags, die einen Inhalt einschließen, auch selbstschließende Tags. Es handelt sich bei diesen Tags entweder um Tags für die Einbindung von externen Ressourcen oder Funktionselemente wie Buttons.

Ein selbstschließender Tag - wie der Name schon sagt - schließt sich selbst d.h. er benötigt keinen eigenen schließenden Tag. Ein Beispiel hierfür ist der Tag für die Ausgabe eines Bilder oder ein Button zum Abschicken eines Formulars. Der Tag wird innerhalb des öffnenden Tags direkt wieder mit einem Slash "/" geschlossen

img /

Seit HTML5 ist es nicht mehr nötig die selbstschließenden Tags auch explizit mit dem Slash zu schließen, da durch die richtige Umsetzung der Spezifikation der nachfolgende Inhalt automatisch wieder ausgeschlossen wird. Auf Grund der Einheitlichkeit oder auch Umsetzungsprobleme von den leider immer noch zu stark vertretenen alten Browsern sollte der Tag trotzdem explizit geschlossen werden - HTML5 stellt trotzdem auch alles richtig dar.

Listing 2.5

```
1 <input type="text" />
2 
```

HTML

### 2.2.1 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 2.6

```

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>

```

HTML

## 2.2.2 Tables

### Table

[Englisch]: a table



Das wohl erste an das man denkt, wenn man das Wort Table hört, ist wohl ein Tisch. Doch was hat dieser Tisch mit **HTML** zu tun?

Natürlich nichts! Jedoch hat das Wort Table im Englischen 2 Bedeutungen:

- Tisch
- Tabelle

Nachdem es uns wohl im Schwerpunkt nicht darum gehen wird, möglichst viele Bilder von Tischen auf unseren Seiten unterzubringen -- außer wir sind ein Schreiner mit Online-Shop -- wenden wir uns dem Thema **Tabellen in HTML** zu. Tabellen in HTML Das folgende Beispiel zeigt den minimalen Aufbau einer Tabelle.

Tabellen bestehen aus Reihen und Zeilen (rows tr und columns td).

Listing 2.7

```

1 <table>
2     <tr>
3         <td>Zelleninhalt</td>
4     </tr>
5 </table>

```

HTML

## Sinnvolle Tabellen

|              |              |              |
|--------------|--------------|--------------|
| Zelleninhalt | Zelleninhalt | Zelleninhalt |
| Zelleninhalt | Zelleninhalt | Zelleninhalt |
| Zelleninhalt | Zelleninhalt | Zelleninhalt |

Listing 2.8

```

1  <table>
2      <tr>
3          <td>Zeileninhalt</td>
4          <td>Zeileninhalt</td>
5          <td>Zeileninhalt</td>
6      </tr>
7      <tr>
8          <td>Zeileninhalt</td>
9          <td>Zeileninhalt</td>
10         <td>Zeileninhalt</td>
11     </tr>
12     <tr>
13         <td>Zeileninhalt</td>
14         <td>Zeileninhalt</td>
15         <td>Zeileninhalt</td>
16     </tr>
17 </table>

```

HTML

## Semantik in Tabellen

im folgenden Beispiel wurde unsere Tabelle um die beiden Element

- thead und
- tbody

ergänzt.

Diese beiden Elemente dienen dazu innerhalb der Tabelle 2 verschiedene Bereiche zu definieren:

- Den Tabellenkopf mit der Beschreibung und
- die Tabelleninhalte

| Kopfzelle 1  | Kopfzelle 2  | Kopfzelle 3  |
|--------------|--------------|--------------|
| Zelleninhalt | Zelleninhalt | Zelleninhalt |
| Zelleninhalt | Zelleninhalt | Zelleninhalt |
| Zelleninhalt | Zelleninhalt | Zelleninhalt |

Listing 2.9

```

1  <table class="example">
2      <thead>
3          <tr>
4              <th>Kopfzelle 1</th>
5              <th>Kopfzelle 2</th>
6              <th>Kopfzelle 3</th>
7          </tr>
8      </thead>
9      <tbody>
10         <tr>
11             <td>Zelleninhalt</td>
12             <td>Zelleninhalt</td>
13             <td>Zelleninhalt</td>
14         </tr>
15         <tr>
16             <td>Zelleninhalt</td>
17             <td>Zelleninhalt</td>
18             <td>Zelleninhalt</td>
19         </tr>
20         <tr>
21             <td>Zelleninhalt</td>
22             <td>Zelleninhalt</td>
23             <td>Zelleninhalt</td>
24         </tr>
25     </tbody>
26 </table>

```

HTML

## TableEnd?

- Finden Sie heraus ob es auch einen Bereich TabellenEnde gibt.  
Falls ja, testen Sie an selbst gewählten Beispielen, wie sich dieser verhält.
- Denken Sie sich ein eigenes Beispiel aus, warum es solch einen Bereich innerhalb einer Tabelle geben sollte.

## Experimente mit Tabellen

|              |                       |              |              |              |              |
|--------------|-----------------------|--------------|--------------|--------------|--------------|
| Kopfzelle 1  | Kopfzelle 2           | Kopfzelle 3  |              |              |              |
| Zelleninhalt |                       |              | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          | Zelleninhalt | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          | Zelleninhalt | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          | Zelleninhalt | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          | Zelleninhalt | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          |              | Zelleninhalt |              |              |
| Zelleninhalt | Zelleninhalt          | Zelleninhalt | Zelleninhalt | Zelleninhalt | Zelleninhalt |
| Zelleninhalt | A table<br>in a table | Zelleninhalt | Zelleninhalt | Zelleninhalt | Zelleninhalt |

## Listing 2.10

```

1  <table class="example">
2      <thead>
3          <tr>
4              <th>Kopfzelle 1</th>
5              <th>Kopfzelle 2</th>
6              <th>Kopfzelle 3</th>
7          </tr>
8      </thead>
9      <tbody>
10         <tr>
11             <td colspan="3" class="green">Zelleninhalt</td>
12             <td>Zelleninhalt</td>
13         </tr>
14         <tr>
15             <td>Zelleninhalt</td>
16             <td>Zelleninhalt</td>
17             <td>Zelleninhalt</td>
18             <td>Zelleninhalt</td>
19         </tr>
20         <tr>
21             <td>Zelleninhalt</td>
22             <td>Zelleninhalt</td>
23             <td>Zelleninhalt</td>
24             <td>Zelleninhalt</td>
25         </tr>
26         <tr>
27             <td rowspan="2" class="green">Zelleninhalt</td>
28             <td>Zelleninhalt</td>
29             <td>Zelleninhalt</td>
30             <td>Zelleninhalt</td>
31         </tr>
32         <tr>
33             <td>Zelleninhalt</td>
34             <td>Zelleninhalt</td>
35             <td>Zelleninhalt</td>
36         </tr>
37         <tr>
38             <td>Zelleninhalt</td>
39             <td colspan="2" rowspan="2" class="green">Zelleninhalt</td>
40         </tr>
41         <tr>
42             <td>Zelleninhalt</td>
43             <td>Zelleninhalt</td>
44         </tr>
45         <tr>
46             <td>Zelleninhalt</td>
47             <td>Zelleninhalt</td>
48             <td>Zelleninhalt</td>
49             <td>Zelleninhalt</td>
50             <td>Zelleninhalt</td>
51             <td>Zelleninhalt</td>
52         </tr>
53         <tr>

```

## Selbststudium

### Aufgabe

Bauen Sie nachfolgende Tabelle inhaltlich korrekt nach

| laufende Nummer | Lehrgang |             | Teilnehmer     |        |
|-----------------|----------|-------------|----------------|--------|
|                 | Nr.      | Bezeichnung | PK             | Name   |
| 1               | 269317   | Java        | 123456-D-78901 | Saturn |
| 2               |          |             | 109876-I-54321 | Venus  |
| 3               |          |             | 209684-N-43829 | Mars   |
| 4               |          |             | 567483-G-78965 | Pluto  |
| 5               | 268318   | C++         | keiner         |        |

## 2.2.3 Headings

### Headings - Überschriften

Überschriften dienen in HTML der Strukturierung der Inhalte. HTML kennt 6 Stufen von Überschriften. Je niedriger die Nummer, desto wichtiger (besser) ist die Überschrift. [Schulnotensystem]

Eine Überschrift H1 stellt somit immer die wichtigste Überschrift auf einer Seite dar.

Eine h1 - Überschrift sollte nur einmalig auf einer Seite vertreten sein. Als Überschrift für genau den Artikel, der auf der Seite zu finden ist.

Eine Suchmaschine wie zum Beispiel Google bewertet die Wichtigkeit des Textes einer Überschrift nach der Klassifizierung mit dem jeweiligen H Tag.

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

#### Heading 5

#### Heading 6

Listing 2.11

|   |   |
|---|---|
| 1 | <code>&lt;h1&gt;Heading 1&lt;/h1&gt;</code> |
| 2 | <code>&lt;h2&gt;Heading 2&lt;/h2&gt;</code> |
| 3 | <code>&lt;h3&gt;Heading 3&lt;/h3&gt;</code> |
| 4 | <code>&lt;h4&gt;Heading 4&lt;/h4&gt;</code> |
| 5 | <code>&lt;h5&gt;Heading 5&lt;/h5&gt;</code> |
| 6 | <code>&lt;h6&gt;Heading 6&lt;/h6&gt;</code> |

HTML

## Hierarchische Strukturierung

Erstellenn nur mittels Überschriften eine HTML-Seite, die die folgende Struktur logisch abbildet:

- HTML
  - Grundlagen
  - Inhalte
    - Tabellen
    - Listen
    - Formulare
      - Radio-Buttons
      - Checkboxes
      - Input
      - Textarea
  - Überschriften
- Struktur
  - Aside
  - Header
  - Footer
  - Navigation

### 2.2.4 Paragraphs

#### HTML ignoriert Umbrüche im Code

Zeilenumbrüche sollten nur innerhalb von Elementen genutzt werden, und nicht für das Layout "missbraucht" werden. Beispiel:

Dies ist ein einzelner Paragraph. Dies ist der zweite Satz des Paragraphs. Dies ist der dritte Satz des Paragraphs

Listing 2.12

```
1 <p>Dies ist ein einzelner Paragraph.
2 Dies ist der zweite Satz des Paragraphs. Dies ist der dritte Satz des Paragraphs</p>
```

HTML

#### Beispiele für Zeilenumbrüche:

Beispiel für Zeilenumbrüche

Dies ist ein einzelner Paragraph.  
Dies ist der zweite Satz des Paragraphs.  
Dies ist der dritte Satz des Paragraphs



Listing 2.13

```
1 <p>Dies ist ein einzelner Paragraph.<br />Dies ist der zweite Satz des Paragraphs.</p>
```

HTML

## Zusammenfassung

- p Ein Paragraph in HTML entspricht dem Tastenkommando "ENTER" in Word.
- br/Ein Zeilenumbruch in HTML entspricht dem Tastenkommando "SHIFT-ENTER" in Word.

## Paragraphen und Zeilenumbrüche

Paragraphen dienen wie überall dazu, einen längeren Text in logische Abschnitte zu unterteilen. Genauso wie bei Word, sollte hierzu NICHT der einfache Zeilenumbruch [WORD: SHIFT-ENTER] genutzt werden. Sondern es sollte dafür ein neuer Paragraph erstellt werden [WORD: ENTER]. In HTML wird dies durch den Tag p realisiert. Der einfache Zeilenumbruch ist der selbstschließende Tag br/

Beispiel

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia nunc eu quam vestibulum pretium. Aliquam ullamcorper turpis ut nisi imperdiet dignissim. Quisque lacinia suscipit justo, ut mattis nisl dignissim ac. Morbi eget nibh eu neque auctor laoreet sed id leo. Suspendisse potenti. Maecenas sed sapien purus. Ut molestie pharetra justo sed hendrerit. Proin id odio vitae diam ullamcorper rhoncus. Nulla facilisi.

Nam non nisl nisl. Vivamus ac pulvinar erat. In hac habitasse platea dictumst. Praesent tincidunt, neque eu pellentesque luctus, risus tortor porttitor sapien, et molestie nibh est sit amet neque. Integer sollicitudin lacus hendrerit risus dictum nec egestas ligula condimentum. In hac habitasse platea dictumst. Praesent ornare magna eu nisl ullamcorper et tincidunt dolor dapibus. Phasellus viverra elit ac nisi semper sed feugiat purus interdum. Nulla facilisi. Phasellus ornare tincidunt facilisis. Sed fringilla, dui adipiscing blandit rhoncus, mi dui mollis erat, sit amet facilisis risus ipsum id ipsum. Mauris et blandit felis. Mauris ligula nunc, cursus et ultricies quis, pharetra in arcu.

Donec suscipit mi sed erat tempus tempus. Duis a magna sapien. Aenean elit nibh, dignissim dignissim vestibulum eu, accumsan quis nibh.

Sed ut nisi eget neque volutpat iaculis vel eu ante.

Sed placerat, augue vitae fringilla placerat, metus neque viverra odio, in laoreet nunc nibh euismod orci. Aliquam eu purus dolor.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam eu ullamcorper dui.

Listing 2.14

```
1 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia nunc eu quam ves
2
3 <p>Nam non nisl nisl. Vivamus ac pulvinar erat. In hac habitasse platea dictumst. Praes
4
5 <p>Donec suscipit mi sed erat tempus tempus. Duis a magna sapien. Aenean elit nibh, digniss
6 Sed ut nisi eget neque volutpat iaculis vel eu ante.<br />
7 Sed placerat, augue vitae fringilla placerat, metus neque viverra odio, in laoreet nunc nibh
8 Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas
```

HTML

## 2.2.5 Images

### Bilder in HTML

Bilder werden in HTML mittels des IMG tag eingebunden. Der Image ist auch ein selbstschließender Tag. Die wichtigsten Attribute, der Img enthalten darf sind:

- src

Hier wird der Pfad des Bildes angegeben, dass eingebunden werden soll. Die Pfadangabe ist relativ von der HTML-Datei. Alternativ kann der Pfad auch absolut angegeben werden.

- alt

Das alt - Attribut gibt eine **alternative** Darstellung an. Falls ein User innerhalb seines Clients die Darstellung von Bildern abgeschaltet hat, wird ihm dieser Text präsentiert. Des weiteren ist diese Beschreibung relevant in Bezug auf Barrierefreiheit. Personen, die einen ScreenReader nutzen, wird an der Stelle des Bildes diese Beschreibung vorgelesen.

- title

Das title - Attribut gibt dem Bild einen Titel. Dieser wird zum Beispiel beim MouseOver angezeigt



Listing 2.15

```
1 Link</a>
```

HTML

## Link zu einer E-Mail Adresse

Mit Hilfe der normalen Links können auch Verknüpfungen zu e-Mail Adressen erstellt werden. Funktionsweise: Ein Klick auf solch einen Link öffnet das im Betriebssystem als Standard eingestellte e-Mail-Programm und übergibt die hinterlegte Email-Adresse. Schick mir eine Mail

Listing 2.17

```
1 | <a href="mailto:test@example.org">Schick mir eine Mail</a>
```

HTML

## Links in neuen Fenstern

Mittels der Attribute eines Links kann eingestellt werden, dass ein Link z.B. in einem neuen Fenster geöffnet werden soll. Dies geschieht durch Angabe des Attributes: target Mögliche Werte für das target Attribut:

| Wert             | Auswirkung  |
|------------------|---|
| _blank           | neues Fenster   |
| _self (standard) | in eigentlichen Fenster                                       |
| _top             | im eigentlichen Fenster (nur bei Nutzung von Frames relevant) |
| _parent          | im Eltern-Frame (nur bei Nutzung von Frames relevant)         |
| framename        | im benannten Frame  |

Beispiel Google in neuem Fenster

Listing 2.18

```
1 | <a href="http://www.google.de" target="_blank">Google in neuem Fenster</a>
```

HTML

## verschiedene Beispiele für Links

Bilder als Links



Container als Links

Ein komplettes DIV als Link

Buttons als Links Button Text als Links Dieser Text ist ein Link

Ein Klick auf den Hund öffnet das Bild in seinem Ursprung

Listing 2.19

```

1 <span class="h4">Bilder als Links</span>
2 <a href="./img/1342449951le-pic.jpg"></a>
3 <div class="gradientbox floatright">Ein Klick auf den Hund Öffnet das Bild in seinem Ursprung</div>
4 <div class="clear"></div>
5 <span class="h4">Container als Links</span>
6 <a href=""><div class="blackbox">Ein komplettes DIV als Link</div></a>
7 <span class="h4">Buttons als Links</span>
8 <a href=""><button class="button">Button</button></a>
9 <span class="h4">Text als Links</span>
10 <a href="">Dieser Text ist ein Link</a>

```

## 2.2.7 Formulare

### Formulare in HTML

Zu Beginn der Zeiten des Internet, war der Schwerpunkt lediglich bei der Bereitstellung von Daten. Der Austausch von Daten und Informationen wurde noch auf den herkömmlichen Transportwegen wie z.B. physikalischen Datenträgern, eMail oder FTP vollzogen. Mit der Einführung von HTML2 im Jahr 1995 wurde die Möglichkeit geschaffen mit HTML Formulare auszugeben. Anfangs wurden diese meist lediglich dazu benutzt etwas Interaktivität auf den Seiten zu schaffen. Es wurden Kontaktformulare angeboten, so dass nicht jedes mal eine angegebene Email Adresse kopiert und in sein eigenes lokales Email-Programm kopiert werden musste. Der User konnte direkt auf der Seite Kontakt mit dem Author aufnehmen.

Diese Idee entwickelte sich dahin weiter, dass es für die User auch möglich sein sollte, Kommentare zu hinterlassen um zusammen auf einfache Weise zu diskutieren - die ersten einfachen Foren war geboren.

Aus dieser Idee heraus - die Inhalte von Webseiten zusammen zu erstellen - kamen dann die ersten Ideen für Content Management Systeme, professionelle Lösungen für Foren, Formulare zum Suchen auf den Seiten, etc.

Mittlerweile ist auf fast keiner Seite mehr nur noch statischer Inhalt angeboten, sondern der User kann überall interagieren - mit **Formularen**.

Egal ob es die Auswahl eines Hotels - das Hinterlassen eines Kommentars, das posten eines Status auf Facebook oder die Suche nach einem Artikel auf Ebay ist - all das sind Formulare.

## form

Ein Formular in HTML wird mittels des Tags `form` realisiert. Jedes Formular muss die Attribute `action` und `method` besitzen. Die beiden Attribute beschreiben, **wohin** und **wie** die Inhalte übergeben werden sollen

Mögliche Werte für die Attribute: `action`

Wert                      Auswirkung

`" "` (deprecated) übergibt die Werte an sich selber (lädt dadurch die Seite neu)

`formular.php` übergibt die Werte an die Datei `formular.php`

`method`

Wert                      Auswirkung

`post` überträgt die Daten im HTTP - Header

`get` überträgt die Daten in der URL - z.B. `http://google.de?suche=FORMULARINHALT`

## Ein erstes Formular

und der dazugehörige Code:

Listing 2.20

```
1 <form action="" method="">
2     <input type="text"/>
3     <input type="submit"/>
4 </form>
```

HTML

## das input element

das `input` Element stellt das meistgenutzte Eingabefeld innerhalb eines Formulars dar. Das `input` Element ist ein variables Element und kann je nach Definition durch das Attribut `type` ein verschiedenes Aussehen haben und verschiedene Daten aufnehmen.

Beispiele:

| Wert                         | Auswirkung                            |
|------------------------------|---------------------------------------|
| <code>type="text"</code>     | <input type="text"/>                  |
| <code>type="password"</code> | <input type="password"/>              |
| <code>type="search"</code>   | <input type="search"/>                |
| <code>type="submit"</code>   | <input type="submit" value="Submit"/> |

Durch setzen des Attributs `maxlength="4"` wird die Eingabe in dieses Feld auf 40 Zeichen begrenzt

## mehrzeilige Eingabebereiche

Wir haben bereits das Element `input` fuer Formulare kennen gelernt. Mit dessen Hilfe war es uns schon möglich ein Formular zu erstellen, um einfache Daten aufzunehmen. Stellen wir uns jedoch vor, wir möchten es dem Nutzer ermöglichen uns ein längeres Kommentar zu übersenden. Das Element `input` stellt uns nur eine einzige Zeile zur Verfügung. Dies macht es offensichtlich sehr schwer einen größeren Text einzutragen und diesen im Überblick zu behalten

Zur Lösung dieses Problems ist in HTML das Element `textarea` vorgesehen, welches einen beliebig großen Eingabebereich zur Verfügung stellt.

Beispiel: Inhalt der Textarea

Mittels der Attribute cols und rows definiert man, wieviele Spalten bzw. Reihen vorgegeben werden sollen. Moderne Browser erlauben zudem, das textarea Element mit der Maus auf die jeweiligen Bedürfnisse des Nutzers temporär zu vergrößern

Listing 2.21

```
1 <textarea cols="75" rows="5">Inhalt der Textarea
```

HTML

## 2.2.8 Kommentare

### Einfache Kommentare

Wie in einer normalen Programmiersprache bietet auch HTML die Möglichkeit den Quellcode zu kommentieren.

Die eingebundenen Kommentare werden dann vom Browser dann nicht im gerenderten Frontend angezeigt

Listing 2.22

```
1
2 <!-- Dies ist ein Kommentar -->
3
4
```

HTML

### Mehrzeilige Kommentare

HTML unterscheidet nicht zwischen ein- und mehrzeiligen Kommentaren.

Ein Kommentar Absatz wird daher genauso erstellt wie ein einzeliger Kommentar.

Listing 2.23

```
1 <!--
2     Dies ist ein mehrzeiliger Kommentar
3
4     Zweite Zeile des Kommentars
5 -->
```

HTML

## 2.2.9 Textformatierung

### Einfache Formatierungen

This text is bold

This text is italic

This is computer output

This is subscript and superscript

Dieser Text ist durchgestrichen

## b vs. strong

In den meisten normalen Browsern werden die beiden TAGs b und strong identisch dargestellt. Sie dürfen jedoch nicht als komplementär angesehen werden.

Unterschiede: b Der Tag b dient lediglich der optische fetteren Darstellung eines Ausdrucks. Inhaltlich (semantisch) hat dieser Ausdruck nicht mehr Bedeutung als normaler Text. strong Der Tag strong hat zusätzlich zur optischen Gestaltung eine semantische Bedeutung: Ein Ausdruck der mit strong markiert wurde ist als wichtiger zu betrachten, als normaler Text. Warum sollte mich das interessieren wenn beide Tags exakt gleich dargestellt werden?

Beispiel:

Ein Screenreader (Gerät, dass den Inhalt einer Homepage vorlesen kann - wird benutzt von Menschen mit Sehbehinderungen) würde bei dem Tag strong diesen Ausdruck auch betont vorlesen. Dies ist bei dem Tag b nicht der Fall.

## em vs. i

Parallel zu den tags b und strong verhalten sich auch die Tags EM und I. EM hat eine semantische Bedeutung und I besitzt lediglich die optischen Eigenschaften für den Browser

## 2.2.10 Listen

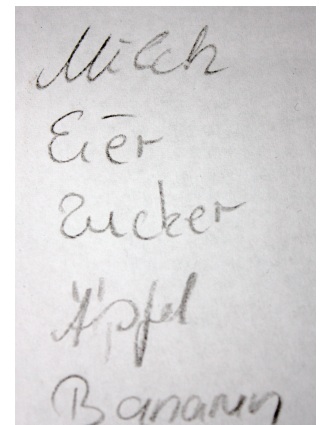
### Geordnete und ungeordnete Listen

| ungeordnete Liste | nummerierte Liste  |
|-------------------|--------------------|
| ▪ Listenelement   | 1. erstes Element  |
| ▪ Listenelement   | 2. zweites Element |
| ▪ Listenelement   | 3. drittes Element |
| ▪ Listenelement   | 4. viertes Element |
| ▪ Listenelement   | 5. fünftes Element |

### Nummerieren, Definieren, oder einfach nur so....

HTML kennt 3 verschiedene Möglichkeiten Listen zu erstellen:

- unsortierte Listen
- sortierte Listen
- Definitionslisten



## Unsortierte Listen - unordered lists [ul]

Die einfachste Form einer Liste ist die Aufzählung.

In HTML wird die unsortierte Liste mit dem Element ul eingeleitet und dementsprechend mit /ul wieder geschlossen.

Die einzelnen Elemente innerhalb einer Liste werden mit dem Tag li definiert.

Gemäß den Spezifikation von HTML ist es nicht nötig, jedes einzelne li auch wieder explizit zu schließen. Um eventuelle Darstellungsprobleme im vornherein zu verhindern und auch einen einheitlichen Coding-Stil zu gewährleisten, sollten alle li auch wieder mit /li geschlossen werden.


- Milch
- Äpfel
- Bananen
- Joghurt

Listing 2.24

```

1 <ul>
2   <li>Milch</li>
3   <li>Äpfel</li>
4   <li>Bananen</li>
5   <li>Joghurt</li>
6 </ul>

```



## 2.3 Struktur in HTML

### 2.3.11 BLOCK

### 2.3.12 Blockelemente vs. Inline Elemente

#### Gruppierung von Elementen

HTML-Elemente können mit Hilfe der beiden Tags span und div zu Gruppen zusammengefasst werden. Der Unterschied zwischen diesen beiden Elementen liegt in Ihrer Darstellungsform.

Grundsätzlich haben diese beiden Elemente keinerlei Auswirkung auf das Design. Jedoch hat ein DIV-Container von Haus aus eine Breite von **100%**. Ein span-Tag hingegen hat keine feste Breite, sondern gliedert sich in den fließenden Tag mit ein.

Ein span sollte also benutzt werden, wenn man nur einzelne Buchstaben oder Wörter innerhalb eines Textes markieren will; ein div hingegen, wenn man einen ganzen Block gruppieren und hervorheben will.

#### HTML Block Elemente

**BLOCK** Elemente werden normalerweise -- (kann mit CSS überschrieben werden) -- in einer neuen Linie dargestellt.

Sie haben grundsätzlich eine **Breite** von **100%**. Beispiele für Block Elemente:

- h1
- p
- ul



- table
- div

### Inhalt des orange P

Listing 2.25

```

1  <style type="text/css">
2      <!--
3          .orange {
4              background:orange;
5          }
6      -->
7  </style>
8
9
10
11 <p class="orange">Inhalt des orange <span class="tag">P</span></p>

```

HTML

## HTML Inline Elemente

HTML Inline-Elemente werden im Gegensatz zu den Block-Elementen normalerweise innerhalb des normalen Textfluss dargestellt. Beispiele für Inline Elemente:

- span
- img
- strong
- em
- a

### Inhalt des orangeSPAN

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae eros eros, vel ultricies dolor. Aenean ultricies adipiscing viverra. Morbi in adipiscing quam. Proin neque lacus, lacinia placerat porta at, adipiscing ut neque. Nunc metus arcu, consectetur id lobortis in, lacinia nec augue. Ut vitae ipsum sit amet velit ornare dapibus. Curabitur eget libero vitae sem hendrerit porttitor id sed arcu. Sed condimentum ligula at nunc suscipit nec auctor tortor interdum. Proin non leo leo, sed consequat leo. Aliquam eu libero mi. Etiam non orci ac turpis porta laoreet. Ut sollicitudin porttitor mi, vel consectetur lorem lobortis ut. Nunc ipsum ipsum, faucibus eget rhoncus ac, faucibus eu magna.

Listing 2.26

```

1  <style type="text/css">
2      <!--
3          .orange {
4              background:orange;
5          }
6      -->
7  </style>
8
9
10 <span class="orange">Inhalt des orange SPAN</span>
11
12
13 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae eros eros, vel ut
14 Aenean ultricies adipiscing viverra. Morbi in adipiscing quam. Proin neque lacus, lacinia
15 adipiscing ut neque. Nunc metus arcu, consectetur id lobortis in, <span class="orange">laci
16 Ut vitae ipsum sit amet velit ornare dapibus. <span class="orange">Curabitur</span> eget l
17 porttitor id sed arcu. Sed condimentum ligula at nunc suscipit nec auctor tortor interdum.
18 consequat leo. Aliquam eu libero mi. Etiam non orci ac turpis porta laoreet. Ut sollicitud
19 lorem lobortis ut. Nunc ipsum ipsum, faucibus eget rhoncus ac, faucibus eu magna.</p>

```

HTML

### 2.3.13 Semantische Strukturierung

#### Bekämpfung des Missbrauchs

Mit der Einführung von HTML5 wurden endlich Elemente geschaffen, um eine Seite semantisch korrekt zu strukturieren. In den Zeiten vor HTML5 wurden meist normale Container dazu missbraucht, mittels Ihrer ID zu beschreiben, was sie beinhalten. Durch die Analyse von Milliarden von Webseiten konnte ein eindeutiges Muster ermittelt werden, welche Bereiche auf einer Webseite zwingend notwendig sind. In fast allen Fällen gab es Container mit den folgenden IDs:

- #nav bzw. #navigation
- #header
- #footer
- #content
- #content\_left bzw. #content\_right

Da bei dieser empirischen Untersuchung derart viele Übereinstimmungen in der Struktur von Webseiten gefunden wurde, veranlasste dies die **WHATWG** (Community des W3C zur Weiterentwicklung von HTML) gewisse Elemente zu schaffen, die exakt nur diesem Zweck dienen. Durch die Nutzung dieser neuen Elemente wird versucht eine größere Vereinheitlichung zu schaffen und zusätzlich den Computern mehr Hilfe zu bieten, die Struktur einer Seite an Hand eindeutiger Objekte zu verstehen. Die wichtigsten neugeschaffenen Elemente sind:

- nav
- header
- footer
- article
- aside

Eine genauere Erklärung dieser Elemente erübrigt sich auf Grund der Eindeutigkeit der Namen.

Diese Elemente verhalten sich in der Handhabung mit CSS exakt wie ein Container. --> auch ältere Browser stellen diese Elemente richtig dar. (nämlich ohne optische Darstellung, so wie einen Container)

Exakte Definition der ElementeWHATWG

## 3 CSS

### 3.1 Schriften



### 3.2 CSS3 - Effekte

#### 3.2.14 text-shadow

##### Einfacher Textschatten

# Einfacher Textschatten

## Breiter Textschatten

Einfacher Textschatten  
Einfacher Textschatten  
Einfacher Textschatten  
Kontur

Letterpress

Emboss

# Glow-Effekt



# Neon

# feuerartiger

## Retro Design

text-shadow

`text-shadow: 1px 1px 5px #333;`

X-Versatz  
Y-Versatz  
Blur  
Farbe

Die text-shadow - Eigenschaft wurde bereits in CSS 2.0 eingeführt. Jedoch wurde diese CSS-Eigenschaft von keinen Browser implementiert. In CSS 2.1 wurde die Textschatten-Eigenschaft entfernt und erst wieder in CSS3 in Kraft gesetzt. Daraufhin fingen fast alle Browser-Hersteller mit der Umsetzung der Text-Schatten an.

Listing 3.1

```
1 h1 {  
2   text-shadow: 1px 1px 5px #333;  
3 }
```

CSS

### 3.2.15 box-shadow

### 3.2.16 border-radius

## 4 WEBSERVER

---

### 4.1 HTTP und das Web

#### 4.1.17 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.

Benutzen Sie Wikipedia, um die folgenden Fragen zu beantworten:

- 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"?

- 
- HF Weidinger - L Siegerth      ITSysAdminFwWebSK      Seite 33 von 75

Ein **URI** ("Uniform Resource Indicator/Identifizier") bezeichnet die vollständige Adresse, um beliebige Daten auf einem Server im Internet zu finden. Ein Beispiel für einen URI könnte sein:

[http://www.fueustgsbw.de:80/intern/service/lehre\\\_und\\\_ausbildung/index.html](http://www.fueustgsbw.de:80/intern/service/lehre\_und\_ausbildung/index.html)

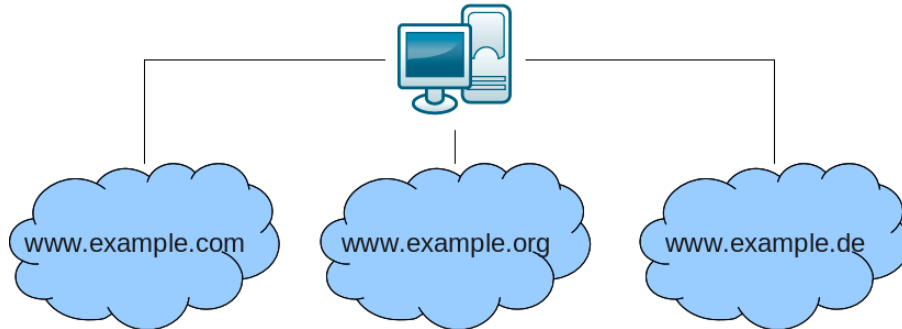
- Wie heißen die Teile, in die sich der obige URI gliedert (Hinweis: Es sind fünf Teile)?
- Worin liegt der Unterschied zwischen einem URI und einem URL?

## 4.2 Apache

### 4.2.18 Virtual Hosting

#### IP-Basiertes Virtual Hossting

Unter dem Begriff "Virtual Hosting" versteht man die Möglichkeit auf einem einzigen physikalischen Webserver mehrere Webangebote gleichzeitig bereitzustellen.



Virtual Hosting kann IP-Basiert oder Namensbezogen erfolgen. Um IP-Basiertes Virtual Hosting betreiben zu können, braucht ein Webserver mehrere IP-Adressen, bzw. eine IP-Adresse mit mehreren freien Ports. Die Reihenfolge der Schritte zur Konfiguration von V-Hosts ist:

1. Konfiguration der IP-Einstellungen am Webserver durch Sys-/Netzwerkadmin.
2. Freischalten von IP-Adresse und Port an einer evtl. vorhandenen Firewall durch den Sys-/Netzwerkadmin.
3. Konfiguration des Apache Webservers mittels des VirtualHost-Kontainers.

Der Kontainer VirtualHost wird verwendet, um in der Hauptkonfigurationsdatei des Apache Webservers einen Virtuellen Host darzustellen. Am Ende dieser Sektion ist ein Beispiel dazu angegeben.

1. Wo können Sie in der Onlinedokumentation ansehen, welche Direktiven für einen VirtualHost-Kontainer zulässig sind?
2. Warum ist hier die Direktive ServerName sehr wichtig? Recherchieren Sie unter: <http://httpd.apache.org/docs/2.4/dns-caveats.html>
3. Benutzen Sie das unten angegebene Konfigurationsbeispiel, um einen Webauftritt für SelfHTML zu erstellen! SelfHTML soll unter der URL <http://localhost:8080> für alle Nutzer erreichbar sein! Welche Konfigurationsschritte sind dazu notwendig?

Für die Angabe eines Ports gibt es drei Möglichkeiten:

- Keine Angabe, z. B. 127.0.0.1: In diesem Fall wird Port 80 angenommen.
- Angabe eines Ports, z. B. 127.0.0.1:8080: Es wird der angegebene Port verwendet.
- Angabe eines \*, z. B. 127.0.0.1:\*: Ein so konfigurierter Virtual Host beantwortet Anfragen auf allen Ports.

[illegible]

Erstellen Sie einen weiteren virtuellen Host, der auf Port 80 den MediaWiki bereitstellt!

Listing 4.1

```
1 <VirtualHost "127.0.0.1:8080">
2     ServerAdmin    root@127.0.0.1
3     DocumentRoot   "E:/apache/httpd/2.4.2/htdocs/port_8080"
4     ServerName      "www.example.org"
5     DirectoryIndex  index.php
6 </VirtualHost>
7
```

Sobald eine größere Anzahl von Virtuellen Hosts auf einem Apache Webserver konfiguriert wird, kann es leicht passieren, dass dessen Konfigurationsdatei sehr unübersichtlich wird. Um dies zu vermeiden, bietet er die Möglichkeit einzelne Konfigurationsabschnitte, z. B. die Konfiguration eines Virtual Hosts, in eine andere Datei auszulagern und diese in die Konfiguration des Servers einzubinden.

Benutzen Sie die Apache Onlinedokumentation, um zu Recherchieren wie die beiden Direktiven Include und IncludeOptional funktionieren, bzw. worin sich beide unterscheiden!

Benutzen Sie für die folgenden Aufgaben die beiden unten angegebenen Virtuellen Hosts!

- Speichern Sie die Konfigurationen der beiden Virtual Hosts in jeweils einer eigenen Datei (port\_80.conf und port\_8080.conf)
- Konfigurieren Sie Ihren Apache so, dass beide VHosts dargestellt werden. Sollte die Konfiguration von VHost 1 nicht gelesen werden können, muss der Start des Apache Servers unterbrochen werden. Bei VHost 2 ist diese Maßnahme nicht notwendig, bzw. nicht erwünscht!

Wenn Sie virtuelle Hosts konfigurieren, sollte der physikalische Webserver höchstens noch eine Fehlerseite bereitstellen. Alle Webauftritte werden dann durch die virtuellen Server dargestellt.

Listing 4.2

```

1  #VHost 1 - port_80.conf
2  <VirtualHost "127.0.0.1">
3      ServerAdmin    root@127.0.0.1
4      DocumentRoot   "E:/Apache/httpd/2.4.2/htdocs/port_80"
5      ServerName     "www.example.org"
6  </VirtualHost>
7
8  #VHost 2 - port_8080.conf
9  <VirtualHost "127.0.0.1:8080">
10     ServerAdmin    root@127.0.0.1
11     DocumentRoot   "E:/Apache/httpd/2.4.2/htdocs/port_8080"
12     ServerName     "www.example.de"
13     DirectoryIndex index.php
14 </VirtualHost>
15

```

PHP

## 4.2.19 Einfuehrung und Installation

### Einführung

Der Apache-Server ist eines der erfolgreichsten freien Software-Projekte in der Geschichte. Man geht derzeit davon aus, dass 54 % aller Websites von einem Apache auf verschiedensten Plattformen ausgeliefert werden. Dabei ist der Apache im Betrieb sehr zuverlässig, performant und durch seine modulare Architektur leicht erweiterbar.

Durch eine große Anzahl von Zusatzmodulen wird der Apache zur geeigneten Plattform für dynamische Inhalte, für sichere Transaktionen oder Workflow-Management. Die Flexibilität im Aufbau und die Vielzahl der unterschiedlichen Erweiterungen bringen eine hohe Komplexität bezüglich der Konfigurationsdateien mit sich, weshalb hier, in dieser Unterrichtsunterlage, nur ein Teil seiner Features und Module beleuchtet werden kann.

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

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



Mit einem Doppelklick können Sie den Apache Service Monitor öffnen.

## Welche Möglichkeiten/Funktionen bietet der Apache Service Monitor?

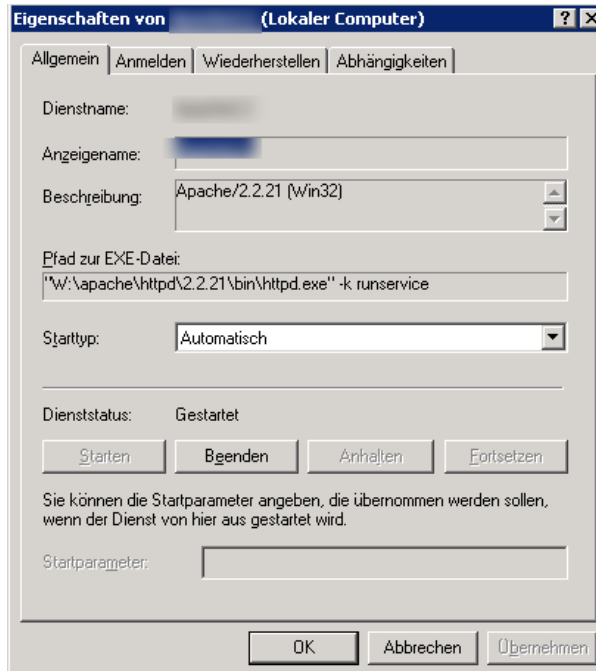
Wenn der Apache HTTP-Server als Dienst installiert wird, ist er selbstverständlich auch in der Windows Dienstkonsole zu finden.

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

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

HF Weidinger - L Siegerth





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





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 Webserver gehören. Die folgende Auflistung zeigt einige davon.

|           |           |
|-----------|-----------|
| Direktive | Bedeutung |
|-----------|-----------|

## ErrorLog

ServerAdmin

ServerName

ServerRoot

Wie bereits zu Anfang erwähnt, ist der Apache HTTP-Server ein modular aufgebautes Produkt. Im Unterverzeichniss `modules`, des Installationsverzeichnis, liegen mehrere Dateien, mit der Dateiendung `.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.

- Seite 45 von 75

- Worin besteht der wesentliche Unterschied zwischen der FallbackResource-Direktive und der Direktive ErrorDocument?
- Konfigurieren Sie Ihren Apache jetzt so, dass er statt der FallbackResource-Direktive die ErrorDocument-Direktive nutzt, um im Falle eines HTTP-404-Fehlers die Fehlerseite zu zeigen!

## 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 4.3

```

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

### 4.2.21 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 Webservers 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.

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

In machen 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:

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

## Benutzung von .htaccess-Dateien





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 4.5

```

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

```

## Zugriffskontrolle durch Autorisierung

### Autorisierung mit Hilfe von Ausdrücken

Das bereits bekannte Kern-Autorisierungsmodul `authz_core_module` stellt den Grundbaustein für alle Autorisierungsmechanismen des Apache HTTP-Server dar. Neben den bereits bekannten Optionen `Require all granted` und `Require all denied` bietet es noch die Möglichkeit Umgebungsvariablen auszuwerten und abhängig vom Ergebnis einer solchen Auswertung den Zugriff zu gewähren.

Nutzen Sie die Apache Onlinedokumentation (<http://httpd.apache.org/docs/2.4/expr.html>), um die folgenden Aufgaben zu erledigen:

- Das Verzeichnis htdocs darf nur zwischen 07:00 Uhr und 16:00 Uhr für die Nutzer verfügbar sein.
- Legen Sie die beiden Verzeichnisse htdocs/msie und htdocs/mozilla an.
- Auf das Verzeichnis htdocs/msie dürfen nur solche Nutzer Zugriff haben, welche einen Internet Explorer als Browser benutzten (Variable HTTP\_USER\_AGENT).
- Auf das Verzeichnis htdocs/mozilla dürfen nur solche Nutzer Zugriff haben, welche den Browser Mozilla Firefox oder Google Chrome benutzen.
- Zugriffe auf das htdocs-Verzeichnis, die mit Hilfe veralteter Browserversionen erfolgen (MSIE 6.x oder Firefox 3.x) sollen direkt abgewiesen werden!

Zusätzlich zu dem Modul authz\_core\_module stellt der Apache noch weitere Module bereit, welche unterschiedlichste Autorisierungsmechanismen zur Verfügung stellen.

|                        |   |
|------------------------|---|
| authnz_ldap_module     | Modul zur Authentifizierung und Autorisierung mittels eines LDAP-Dienstes, z. B. Microsoft Active Directory |
| authz_dbd_module       | Ermöglicht die Autorisierung mit Hilfe einer SQL-fähigen Datenbank z. B. MySQL                              |
| authz_dbm_module       | Dieses Modul benutzt Nutzernamen und Passwörter, welche in DBM-Dateien gespeichert sind.                    |
| authz_groupfile_module | Liest Informationen über Gruppenzugehörigkeiten aus Klartextdateien und benutzt diese zur Autorisierung.    |
| authz_host_module      | Autorisierung mittels IP-Adresse oder Hostname  |
| authz_owner_module     | Ermöglicht die Zugriffssteuerung auf Dateien mit Hilfe der Besitzrechte.                                    |
| authz_user_module      | Ein Autorisierungsmodul, welches basierend auf dem Nutzernamen eine Autorisierung durchführt.               |

## Autorisierung mittels IP-Adresse/Hostname

authz\_host\_module stellt die Möglichkeit bereit eine Autorisierung mittels IP-Adresse oder Hostname durchzuführen.

authz\_host\_module ist kein Ersatz für eine Firewall!

- Kommentieren Sie die Zugriffsregeln aus der letzten Übung aus!
- Konfigurieren Sie den Zugriff auf das htdocs-Verzeichnis so, das ihr eigener Client (Windows 7) keinen Zugriff mehr hat.
- Ändern Sie die Autorisierung für das htdocs-Verzeichnis so, das nur noch lokale Verbindungen zugreifen dürfen!
- Verboten Sie jetzt alle Verbindungen aus Ihrem eigenen IP-Subnetz (10.2.11.64/26)!
- Verändern Sie die Autorisierung des htdocs-Verzeichnisses erneut, so das alle Computer, die Mitglied in der Domäne IT-TRAINING.FUS sind, zugriff haben.

Richten Sie die folgenden Zugriffsregeln ein:

Seite 51 von 75

Listing 4.6

```
1 # Konfiguration 1
2   Require all denied
3   Require local
4
5 # Konfiguration 2
6   <RequireAny>
7     <RequireAll>
8       Require %{TIME_HOUR} -gt 7
9     </RequireAll>
10  </RequireAny>
11  Require local
```

## Abwärtskompatibilität

Im Apache HTTP-Server wurden bis zur Version 2.2 die Direktiven Allow From und Deny From, bzw. Order, statt der Require-Direktive eingesetzt. Um abwärtskompatibel zu bleiben, hat der Apache in der Version 2.4 ein Modul namens mod\_access\_compat. Dies enthält die oben genannten Direktiven, so dass diese auch weiterhin noch eingesetzt werden können.

### 4.2.22 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?



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.

Seite 54 von 75

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

## Seite 55 von 75

Nehmen Sie die Direktiven aus `ispiel{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!



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

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.

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

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

- 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.

```
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
```

## Was sind Server Side Includes?

---

HF Weidinger - L Siegerth

ITSysAdminFwWebSK

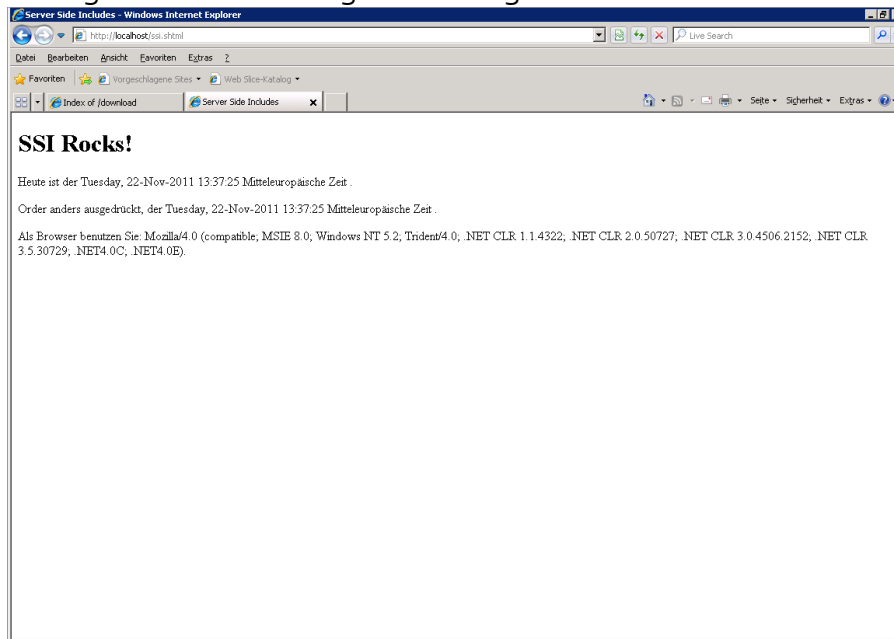
Seite 60 von 75

- Fügen Sie die beiden Direktiven `AddType` und `AddOutputFilter` Ihrer Hauptkonfigurationsdatei hinzu! Was bedeuten sie?
- Laden Sie das Modul `include_module`!

Listing 4.10

Seite 62 von 75

Ein Zugriff auf die obige HTML-Datei zeigt dieses Ergebnis:



- Testen Sie den Zugriff auf ssi.shtml !
- Zusätzlich zur Option Includes gibt es noch die Option IncludesNOEXEC ! Worin unterscheiden sich die beiden?

Listing 4.11

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2      "http://www.w3.org/TR/html4/strict.dtd">
3  <html>
4      <head>
5          <title>Server Side Includes</title>
6      </head>
7      <body>
8          <h1>SSI Rocks!</h1>
9          <p>Heute ist der <!--#echo var="DATE_LOCAL" -->.</p>
10         <!--#config timefmt="%d.%m.%Y" -->
11         <p>Oder anders ausgedr ckt, der <!--#echo var="DATE_LOCAL" -->.</p>
12         <p>Als Browser benutzen Sie: <!--#echo var="HTTP_USER_AGENT" -->.</p>
13     </body>
14 </html>
15

```

HTML

## 4.2.24 CGI

### PERL over CGI

Das Common Gateway Interface stellt eine Schnittstelle zwischen dem Apache HTTP-Server, und externen Programmen, die den Bildschirminhalt generieren, dar. Diese externen Programme werden auch oft als CGI-Programme oder CGI-Skripte bezeichnet. Mit CGI kann auf eine sehr einfache Art und Weise dynamischer Content erzeugt werden.

Ein Zitat aus <http://de.wikipedia.org/wiki/PERL>

Perl ist eine freie, plattformunabhängige und interpretierte Programmiersprache (Skriptsprache), die mehrere Programmierparadigmen unterstützt.

Entworfen worden ist diese Sprache 1987, von Larry Wall, einem Linguisten. Sie sollte als Werkzeug zur Analyse und Verarbeitung von Textdateien dienen. Da sie seit vielen Jahren eine weite Verbreitung in der Welt des Internets gefunden hat, soll sie auch hier, für die Demonstration von CGI-Programmen dienen.

Im Internet existieren zwei große Distributionen von Perl:

- Strawberry Perl
- Active Perl

Bei Active Perl handelt es sich um die Perl-Variante der Firma Activestate. Es existiert in den folgenden Versionen:

- Community Edition
- Business Edition
- Enterprise Edition

Lediglich die Community Edition ist als kostenlose Version erhältlich, darf jedoch nur zu Testzwecken und für den nicht kommerziellen Einsatz genutzt werden. Strawberry Perl hingegen ist eine Open Source Implementierung der Sprache Perl, die ohne Einschränkungen durch eine Lizenz genutzt werden darf. Im folgenden wird in dieser Unterrichtsunterlage Strawberry Perl zum Einsatz kommen.

Laden Sie Strawberry Perl in der aktuellsten Version aus dem Internet herunter und installieren Sie es! Als Installationsverzeichnis soll E:perlstrawberry5.14.2.1 dienen!

## CGI mit ScriptAlias konfigurieren

Um den Apache HTTP-Server dazu überreden zu können das Common Gateway Interface zu aktivieren, ist es notwendig das cgi\_module (mod\_cgi.so) zu laden. Da es sich bei CGI-Skripten um ausführbare Programme handelt, sollten diese nicht mit anderen Dateien, wie z. B. HTML-Dateien vermischt, sondern in einem separaten Verzeichnis aufbewahrt werden. Um dieses Verzeichnis für Nutzer zugänglich zu machen, und gleichzeitig die Ausführung der CGI-Programme zu erlauben, wird die ScriptAlias-Direktive verwendet.

1. Schlagen Sie die Syntax der ScriptAlias-Direktive in der Onlinedokumentation nach!
2. Versehen Sie das Verzeichnis [ServerRoot]/cgi-bin mit dem ScriptAlias cgi-bin Hinweis: [ServerRoot] stellt einen Platzhalter für Ihr Serverroot-Verzeichnis, z. B. E:/apache/httpd/2.4.2, dar!
3. Welcher Schritt muss noch unternommen werden, damit CGI-Skripte aus dem Verzeichnis cgi-bin ausgeführt werden können?

Listing 4.12

```
1 LoadModule cgi_module "modules/mod_cgi.so"
2
```



## CGI mit einem Alias konfigurieren

Eine andere Möglichkeit CGI zu aktivieren besteht darin, dass die Option ExecCGI, in Zusammenhang mit einem Verzeichnisalias und einem Actionhandler genutzt wird.

Legen Sie das Verzeichnis E:/apache/httpd/2.4.2/cgi-bin\_2 an!

Analog zur ersten Variante, muss statt einem ScriptAlias, ein Alias angelegt werden. Im zweiten Schritt, ist es notwendig einen Verzeichniskontainer für das cgi-bin\_2-Verzeichnis anzulegen.

Zeile zwei, AddHandler cgi-script .cgi .pl bewirkt, dass alle Dateien, mit den Endungen .cgi oder .pl als CGI-Skripte betrachtet und ausgeführt werden. Zeile Nummer 7, Options ExecCGI hat zur Folge, dass die Ausführung von CGI-Skripten erlaubt wird. Durch Entfernung dieser Option kann die Ausführung von CGI-Skripten unterbunden werden, ohne das die restliche Konfiguration dazu verändert werden müsste.

Listing 4.13

```

1  Alias "/cgi-bin_2" "E:/apache/httpd/2.4.2/cgi-bin_2"
2
3  <Directory "E:/apache/httpd/2.2.21/cgi-bin_2">
4      #Hinzufuegen eines Actionhandlers fuer CGI
5      AddHandler      cgi-script .cgi .pl
6
7      AllowOverride None
8      Require all granted
9
10     #Die Ausfuehrung von CGI-Skripten zulassen
11     Options          ExecCGI
12 </Directory>
13
```

## Jetzt wird es spannend - Das erste Perl-Programm

Wenn es so etwas wie eine Tradition im Bereich der Programmierung gibt, dann die, dass das erste Programm eines jeden Programmierers die Worte "Hallo Welt!" auf dem Bildschirm ausgibt. Da diese Unterrichtsunterlage keinesfalls mit dieser Tradition brechen möchte, wird sie auch hier fortgesetzt:

- Erstellen Sie das unten abgebildete PERL-Programm und speichern Sie es als Datei, im Verzeichnis E:/apache/httpd/2.4.2/cgi-bin, mit dem Namen hallo\_welt.pl ab.
- Testen Sie das Programm, in dem Sie folgende URL aufrufen:

http://localhost/cgi-bin/hallo\_welt.pl

Es sollte der Schriftzug "Hallo Welt!" erscheinen!

Das Programm hallo\_welt.pl ist wie folgt aufgebaut:

Die erste Zeile ist eine spezielle Kommentarzeile, zu erkennen an den Zeichen #!. Sie hat die Aufgabe, dem Browser mitzuteilen, wo sich der Kommandointerpreter, in diesem Falle die Datei perl.exe, befindet.

Die zweite Zeile benutzt das print-Kommando, um den Contenttype und den für die Ausgabe zu verwendenden Zeichensatz an den Browser weiterzugeben. Der Browser unterdrückt die Ausgabe dieses Textes auf dem Bildschirm, verarbeitet aber trotzdem die Informationen.

Die dritte Zeile ist die alles entscheidende. Sie sorgt für die Bildschirmausgabe von "Hallo Welt!".

Listing 4.14

```
1 #!"E:/perl/strawberry/5.14.2.1/perl/bin/perl.exe"  
2 print "Content-type: text/plain; charset=iso-8859-1";  
3 print "Hallo Welt!";  
4
```

## Die Spannung steigt - Das zweite Perl-Programm

Das zweite Versuch mit CGI ein PERL-Programm aufzurufen soll etwas komplexer gestaltet werden. Im ersten Schritt wird eine HTML-Datei erstellt, die das PERL-Programm mittels eines SUBMIT-Buttons aufruft.

Erstellen Sie die unten abgebildete HTML-Datei begruessung.html und legen Sie sie im Verzeichnis htdocs ab!

Das Beispiel enthält drei Tags, die für die Dateneingabe und -Verarbeitung zuständig sind:

`<input type="text" name="name">` erzeugt ein Textfeld, in das der Nutzer einen beliebigen Freitext eintragen kann. In unserem Beispiel soll hier der Vor-/Nachname eines Benutzers eingetragen werden.

`<input type="submit" value="Begrüßen">` erstellt einen Button mit der Aufschrift "Begrüßen". Hierbei handelt es sich um einen sogenannten "SUBMIT-Button", was bedeutet, dass beim Drücken des Buttons alle Daten aus dem HTML-Formular an ein Ziel übertragen werden. In diesem Falle ist das PERL-Skript `hallo_formular.pl` das Ziel.

`<form action="cgi-bin/hallo_formular.pl" method="POST">` legt fest, dass durch das Drücken eines SUBMIT-Buttons die HTTP-Methode POST genutzt wird, um alle Formulardaten an das PERL-Skript `hallo_formular.pl` zu übertragen.

Erstellen Sie die Datei `hallo_formular.pl` und speichern Sie sie im Verzeichnis `cgi-bin_2` ab. Testen Sie ob HTML-Formular und PERL-Skript funktionieren!

Listing 4.15



```

1  #begrueßung.html
2  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
3      "http://www.w3.org/TR/html4/strict.dtd">
4  <html>
5      <head>
6          <title>Ein HTML-Formular mit PERL</title>
7      </head>
8      <body>
9          <h1>Sei gegrÄsst!</h1>
10         <form action="cgi-bin/hallo_formular.pl" method="POST">
11             <div>
12                 <p>Sag mir deinen Namen!</p>
13                 <input type="text" name="name">
14             </div>
15             <div>
16                 <input type="submit" value="BegrÄssen">
17             </div>
18         </body>
19     </html>
20
21  #hallo_formular.pl
22  #!"E:/perl/strawberry/5.14.2.1/perl/bin/perl.exe"
23
24  print "Content-type: text/plain; charset=iso-8859-1
25
26  ";
27
28  read (STDIN, $query, $ENV{CONTENT_LENGTH});
29
30  print ("Hallo ");
31  print (split(/./, $query, 0));
32  print ("", wie geht es Dir?");
33

```

## 4.2.25 PERL mit mod\_perl.so

### Konfigurieren von mod\_perl.so

Eine zweite Variante PERL in den Apache HTTP-Server einzubinden, stellt das Modul mod\_perl.so dar. Dieses Modul sorgt genauso wie mod\_cgi.so für die Ausführung von PERL-Programmen, aber mit dem Vorteil, dass es nur PERL-Programme, und keine anderen Skripte/Programme zulässt. Dadurch kann verhindert werden, das "aus Versehen" andere Programme, wie z. B. PHP- oder C++ Programme ausgeführt werden.

Als erstes muss die Installation von mod\_perl.so erfolgen. Diese geschieht mit dem PERL eigenen Installationsprogramm "pip" (Perl Installation Program).

1. Öffnen Sie eine Windows-Command-Shell (cmd.exe)

2. Setzen Sie, mit Hilfe des set-Kommandos, die Umgebungsvariable HTTP\_PROXY auf den folgenden Wert:

```
http://username:passwort@192.168.4.250:3128
```

3. Führen Sie das folgende PIP-Kommando aus, um zusätzliche Dateien für das Modul mod\_perl.so zu installieren

```
pip http://strawberryperl.com/package/kmx/mod_perl/5.12_x64/  
mod_perl-2.0.4-MSWin32-x86-multi-thread-5.12.par
```

4. Führen Sie das folgenden PIP-Kommando aus, um zusätzliche Dateien für das das Modul libapreq zu installieren:

```
pip http://strawberryperl.com/package/kmx/mod_perl/5.12_x86/  
libapreq2-2.12-MSWin32-x86-multi-thread-5.12.par
```

5. Kopieren Sie die folgenden Dateien in das Verzeichnis [ServerRoot]/modules:

- mod\_perl.so
- mod\_apreq2.so
- libapreq2.dll
- Konfigurieren Sie den Apache so, dass die folgenden beiden Module geladen werden:
  - perl\_module (mod\_perl.so)
  - apreq\_module (mod\_apreq2.so)
- Legen Sie das Verzeichnis [ServerRoot]/perl an.
- Kopieren Sie das PERL-Skript "hallo\_formular.pl" in das Verzeichnis perl, und entfernen Sie die erste Zeile (Kommentarzeile) aus dieser Datei.
- Passen Sie die Datei begruessung.html so an, dass das PERL-Skript aus dem Verzeichnis perl genutzt wird.
- Konfigurieren Sie in Ihrer Hauptkonfigurationsdatei das Verzeichnis perl so, wie in unten angegeben!
- Legen Sie einen Alias für das Verzeichnis perl an!
- Starten Sie den Apache Webserver neu!
- Testen Sie, ob begruessung.html funktioniert!

Hier werden drei Direktiven verwendet, die für die Ausführung von PERL-Programmen verantwortlich sind. Zwei davon, sind neu.

Die Direktive PerlHandler ModPerl::Registry stellt die Verbindung zwischen mod\_perl.so und dem PERL-Skript her, in dem ein sogenannter "Perlhandler" konstruiert wird. Dieser ermöglicht es mod\_perl.so dann, das Skript aufzufinden und auszuführen.

Die zweite Direktive ist PerlSendHeader On. Sie sorgt dafür, dass mod\_perl.so jedes PERL-Programm nach HTTP-Header-Zeilen durchsucht und diese ausführt. Im Falle des vorangegangenen HalloWelt-Beispiels ist dies die Zeile:

```
print "Content-type: text/plain;charset=iso-8859-1 ";
```

Testen Sie was passiert, wenn Sie die Direktive PerlSendHeader auf den Wert "Off" einstellen!

Listing 4.16

```
1 <Directory "E:/apache/httpd/2.4.2/perl">
2     Options          ExecCGI
3     PerlHandler      ModPerl::Registry
4     PerlSendHeader   On
5     AddHandler       perl-script .pl
6     Require          all granted
7 </Directory>
8
```

## 4.2.26 Apache und PHP

### PHP installieren und konfigurieren



Eine weitere Programmiersprache die im Verlauf der Jahre große Verbreitung im Internet gefunden hat ist PHP. Die Abkürzung PHP stellt ein rekursives Akronym (Ein Akronym ist dann rekursiv, wenn das Akronym selbst in der Bedeutung vorkommt!), mit der Bedeutung "PHP Hypertext Preprocessor" dar. Erschaffen wurde PHP im Jahr 1995 von Rasmus Lerdorf, der es später mit den beiden Softwareentwicklern Andi Gutmans und Zeev Suraski gemeinsam weiterentwickelte.

Im Laufe seiner Entwicklung wurde PHP stark von den Sprachen Perl, C, C++ und Java beeinflusst, weshalb an einigen Stellen die Syntax eine hohe Ähnlichkeit mit diesen Sprachen hat. Da PHP auf fast 75 % aller Webseiten zur Erzeugung/Darstellung von dynamischen Inhalten genutzt wird, ist es zum de facto Standard im Internet geworden. Aktuell läuft die Entwicklung an der Version 5.4.4 (Stand 14.06.2012).

Unter der Adresse <http://windows.php.net/download/> können alle aktuellen PHP-Versionen für Microsoft Windows heruntergeladen werden.

- Laden Sie PHP 5.4.4 VC9 x86 Thread Safe als Zip-Datei herunter!
- Entpacken Sie die Zip-Datei in das Verzeichnis E:\PHP
- Legen Sie eine Kopie der Datei php.ini-production an und benennen Sie diese um in: php.ini!
- Laden Sie die Datei php5apache2\\_2.dll als Modul für Apache!
- Fügen Sie in Ihrer Hauptkonfigurationsdatei, dem Verzeichniskontainer für htdocs, die Direktive AddHandler application/x-httpd-php .php hinzu!
- Starten Sie den Apache neu!
- Erstellen Sie die unten angegebene PHP-Datei, und speichern Sie diese mit dem Namen index.php im Verzeichnis [ServerRoot]/htdocs!
- Rufen Sie die Datei index.php

im Browser auf!

Listing 4.17

```

1  <?php
2      phpinfo();
3  ?>

```

PHP

## PHP + MySQL + Mediawiki

Diese Sektion soll ein einfaches und doch realastisches Beispiel dafür geben, welchen Einsatzzweck PHP in der Praxis hat. Mit Hilfe des Datenbankmanagementsystems MySQL und der PHP-Software MediaWiki wird auf dem Apache Webserver ein kleines Wiki-System erzeugt. Da an dieser Stelle der Einsatz von PHP im Vordergrund steht, werden die Installationen von MySQL und MediaWiki nur rudimentär erläutert.

### MySQL herunterladen und installieren

MySQL ist ein relationales Datenbankmanagementsystem, das als Grundlage für sehr viele dynamische Webauftritte dient. Ursprünglich wurde es von dem schwedischen Unternehmen MySQL AB entwickelt, dass jedoch im Jahr 2008 von Sun Microsystems aufgekauft wurde. 2010 wurde die Firma Sun Microsystems schließlich von Oracle aufgekauft.

MySQL existiert sowohl als kommerzielle Software, als auch als Open Source Produkt, in der Community Edition. Diese steht unter dem folgenden Link zum Download zur Verfügung:

<http://www.mysql.de/downloads/mysql>

Laden Sie MySQL, als Windows MSI-Installer von der obigen Adresse herunter und installieren Sie es mit den folgenden Angaben!

- Installationstyp: Complete
- Haken setzen bei: Launch the MySQL Instance Configuration Wizard
- Detailed Configuration
- Server Machine
- Multifunctional Database
- InnoDB Tablespace Settings: E:MySQLDataFiles
- Online Transaction Processing (OLTP)
- Haken setzen bei: Enable TCP/IP Networking (Port: 3306)
- Haken setzen bei: Enable Strict Mode
- Zeichensatz: Standard Character Set
- Haken setzen bei: Install as Windows Service (Servicename: MySQL)
- Haken setzen bei: Launch the MySQL Server automatically
- Haken setzen bei: Include Bin Directory
- Haken setzen bei: Modify Security Settings (root-Passwort: password)

Nach erfolgter Installation sollte MySQL als Windows Dienst, unter dem Namen glqq MySQLgrqq laufen.

### MediaWiki installieren

MediaWiki wird unter <http://www.mediawiki.org/wiki/Download/de> als Archivdatei, im Format \*.tar.gz, zum herunterladen zur Verfügung gestellt.

- Laden Sie MediaWiki von der obigen Adresse herunter und entpacken Sie den Inhalt des Archives nach E:apachehttpd2.4.2htdocs, so dass dort das Verzeichnis mediawiki-1.x.x entsteht!
- Benennen Sie das Verzeichnis mediawiki-1.x.x in mediawiki um!

Bevor die Installation des MediaWiki gestartet werden kann, müssen noch einige Einstellungen für die Nutzung von PHP vorgenommen werden. PHP wird mit Hilfe der Datei `php.ini` konfiguriert.

Öffnen Sie die Datei `E:\php\php.ini`, und nehmen Sie die im folgenden beschriebenen Änderungen vor!

- Suchen Sie die Zeile `;extension_dir="ext"` und entfernen Sie das Kommentarzeichen (;)!
- Suchen Sie die Zeile `;extension=php_mysql.dll` und entfernen Sie das Kommentarzeichen!
- Speichern und schließen Sie die Datei.
- Kopieren Sie die Datei `E:\php\php.ini` nach `C:\Windows`

ODER

- Setzen Sie die Umgebungsvariable `PHPRC` mit dem Wert: `E:\php`
- Starten Sie die Installation von Mediawiki, in dem Sie die Adresse `http://localhost/mediawiki/index.php` aufrufen!
- Klicken Sie den Link "Please set up the wiki first" an.
- Installieren Sie MediaWiki mit den folgenden Konfigurationseinstellungen:
  - Sprache: de - Deutsch
  - Sprache des Wikis: de - Deutsch
  - Datenbankserver: localhost
  - Datenbankname: my\_wiki
  - Datenbanktabellenpräfix:
  - Name des Datenbankbenutzers: root
  - Passwort des Datenbankbenutzers: password
  - Haken entfernen bei: Dasselbe Konto wie während des Installationsvorgangs verwenden
  - Name des Datenbankbenutzers: mediawiki
  - Passwort des Datenbankbenutzers: password
  - Haken setzen bei: Sofern nicht bereits vorhanden, muss nun das Konto erstellt werden
  - Speicher-Engine: InnoDB
  - Datenbankzeichensatz: UTF-8
  - Name des Wikis: my\_mediawiki
  - Name des Projektnamensraums: Entspricht dem Namen des Wikis
  - Administratorkonto - Name: wikiadmin
  - Administratorkonto - Passwort: passwordwiki
  - Administratorkonto - E-Mail-Adresse:
  - Option: Nein, das Wiki soll nun installiert werden

Befolgen Sie die weiteren Anweisungen auf dem Bildschirm!

## 4.2.27 Die Log-Dateien des Apache

### Das Error-Log

#### Das Error-Log

Ein ganz wesentlicher Aspekt bei Verwaltung eines Serverproduktes, wie z. B. eines Webserver ist es, Informationen über das Verhalten des Servers und über die Zugriffe auf den Server gewinnen zu können. Der Apache HTTP-Server bietet hierzu ein umfassendes Logging an.

Das Error-Log ist die zentrale Logdatei des Apache Webserver. Hier laufen alle Fehler- und Diagnosemeldungen auf, die während des Serverbetriebs entstehen. Unter Microsoft Windows heißt die betreffende Datei standardmässig `error.log` und liegt im Verzeichnis `[ServerRoot]/logs`.



Der Inhalt dieser Datei sollte in regelmässigen Zeitabständen immer wieder kontrolliert bzw. im Fehlerfall als erstes geprüft werden.

Die Meldungen im Error-Log haben ein festes Format, das durch den Admin nicht geändert werden kann. Ein Beispiel für eine Fehlermeldung könnte so aussehen:

```
[Mon Nov 28 08:57:37 2011] [error] [client 127.0.0.1]
Premature end of script headers: login.pl
```

Der erste Teil der Meldung ist das Datum, an dem die Meldung aufgelaufen ist. Der zweite Teil ([error]) stellt den Schweregrad der Fehlermeldung dar. Der dritte Teil gibt die Adresse des Clients an, der durch einen Serverzugriff den Fehler ausgelöst hat. Als letztes kommt schließlich die eigentliche Fehlermeldung.

Konfiguriert wird das Error-Log mit Hilfe der Direktive ErrorLog. ErrorLog "logs/error.log"  
Auch wenn die ErrorLog-Direktive nicht angegeben wurde erstellt der Apache eine Log-Datei namens error.log, im Verzeichnis [ServerRoot]/logs.

## Access Logs

Zusätzlich zum Error-Log kennt der Apache sogenannte Access-Logs. Diese enthalten verbindungspezifische Informationen zu den Fehlermeldungen im Error-Log. Das Format einer Access-Log Datei ist frei wählbar. Ein Access-Log kann mittels der Direktive CustomLog erstellt werden.

Es ist erlaubt mehrere CustomLog-Direktiven in der Hauptkonfigurationsdatei zu verwenden, um mehrere Access-Logs zu konfigurieren.

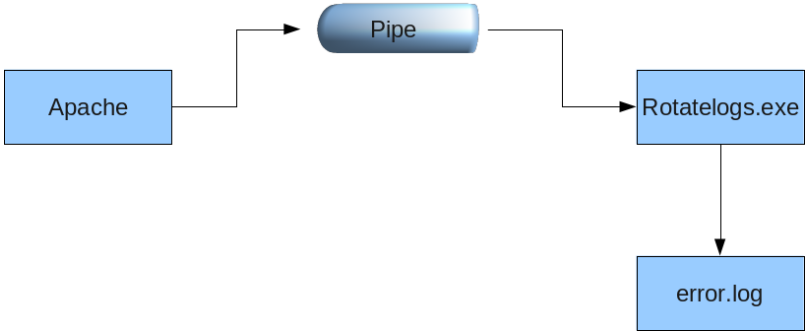
- Wie lautet die Syntax der Direktive CustomLog?
- Welche Aufgabe hat die Direktive LogFormat?
- Wo finden Sie in der Onlinedokumentation Informationen über die einzelnen Formatparameter der Direktive LogFormat?
- Was verbirgt sich hinter der Abkürzung CLF?
- Wie kann ein Format für ein bestimmtes Access-Log festgelegt werden (Syntax)?

## Rotierende Log-Dateien (Log rotation)

Bei einem Webserver auf den täglich mehrere tausend Zugriffe erfolgen, kann eine Log-Datei sehr schnell anwachsen und dadurch unübersichtlich werden. Um so etwas zu vermeiden, kann der Apache seine Log-Dateien in konfigurierbaren Zeitabständen automatisch austauschen. Dieser Mechanismus wird als "Log Rotation" bezeichnet und ist sowohl für das Error-Log als auch für alle Access-Logs getrennt einstellbar.

Sowohl bei der ErrorLog- als auch bei der CustomLog-Direktive sind zwei neue Angaben hinzugekommen. Der erste Teil ist |bin/rotatelog.exe . Der senkrechte Strich, am Anfang der Zeile wird als "Pipe-Zeichen" bezeichnet. Er sorgt für das Pipelining der Logging-Informationen an das Programm rotatelog.exe , welche dann die Meldungen in die jeweilige Log-Datei schreibt, und alle n-Sekunden (hier ist n = 21600) eine neue Datei beginnt.

Unter dem Begriff "Pipelining" versteht man eine Art der Interprozesskommunikation. Das heißt zwei Prozesse tauschen Informationen untereinander aus. In diesem speziellen Fall gibt der Apache Webserver Informationen an das Programm rotatelog.exe weiter.



Konfigurieren Sie Ihr Error-Log so, dass es alle 30 Sekunden rotiert!

[illegible]

Listing 4.18

```
1 ErrorLog "|bin/rotatelog.exe logs/error.log 21600"
2 LogFormat "%h %l %u %t \"%r\" %>s %b" common
3 CustomLog "|bin/rotatelog.exe logs/access.log 21600" common
```

## Bedingtes Logging - Das setenvif\_module-Modul

Um schon während des Loggings eine Auswahl treffen zu können, welche Informationen protokolliert werden sollen, kennt der Apache die Möglichkeit, mit Hilfe des Moduls `mod_setenvif` so Bedingungen für das Logging zu formulieren.

Recherchieren Sie mit Hilfe der Onlinedokumentation nach der Bedeutung der Anweisungen aus dem unten angegebenen Beispiel!

```
1 LoadModule setenvif_module "module/mod_setenvif.so"
2 SetEnvIf Remote_Addr "127.0.0.1" dontlog
3 CustomLog "logs/access.log" env!=dontlog
4
```

Die LogFormat-Direktive erlaubt es, Log-Format für bestimmte HTTP-Fehlercodes festzulegen. Im unten gezeigten Beispiel 1 wird eine Access-Log Datei namens logs/access\_http400error.log erstellt, in welche nur Einträge für den HTTP-Fehlercode 404 - Not Found - gemacht werden. Beispiel 2 hingegen zeigt die Möglichkeit der Verneinung. In der Datei logs/access\_referer.log werden alle Fehlercodes, außer 200, 302 und 304 geloggt.

```
1 #Beispiel 1
2 LogFormat "%404{User-agent}i %U %a" http404
3 CustomLog "logs/access_http404error.log" http404
4
5 #Beispiel 2
6 LogFormat "%!200, 302, 304 {Referer}i" referer
7 CustomLog "logs/access_referer.log" referer
```

Seite 75 von 75