Eine Einführung in den Apache Webserver der Apache Software Foundation

Computerlabor KuZeB

09. März 2009

Geschichte

- Entstanden aus dem “public domain http daemon” von Rob McCool, am NCSA (National Center for Supercomputing Applications) entwickelt.
- 1994 verliess Rob McCool die NCSA - Entwicklung eingestellt
- Es entstand ein Wildwuchs an Bugfixes und Erweiterungen, dies führte zum allgemeinen Bedürfnis einer allgemeinen Distribution
- Ende Februar 1995 formierten sich 8 Leute zur Apache Gruppe
- NCSA httpd 1.3 bildete die Basis für die erste öffentliche Version (0.6.2) des Apache Webserver
- Mai – Juni 1995 entwickelte Robert Thau eine neue Architektur “Shambhala”: modulare Struktur
- August 1995: Apache 0.8.8

Geschichte

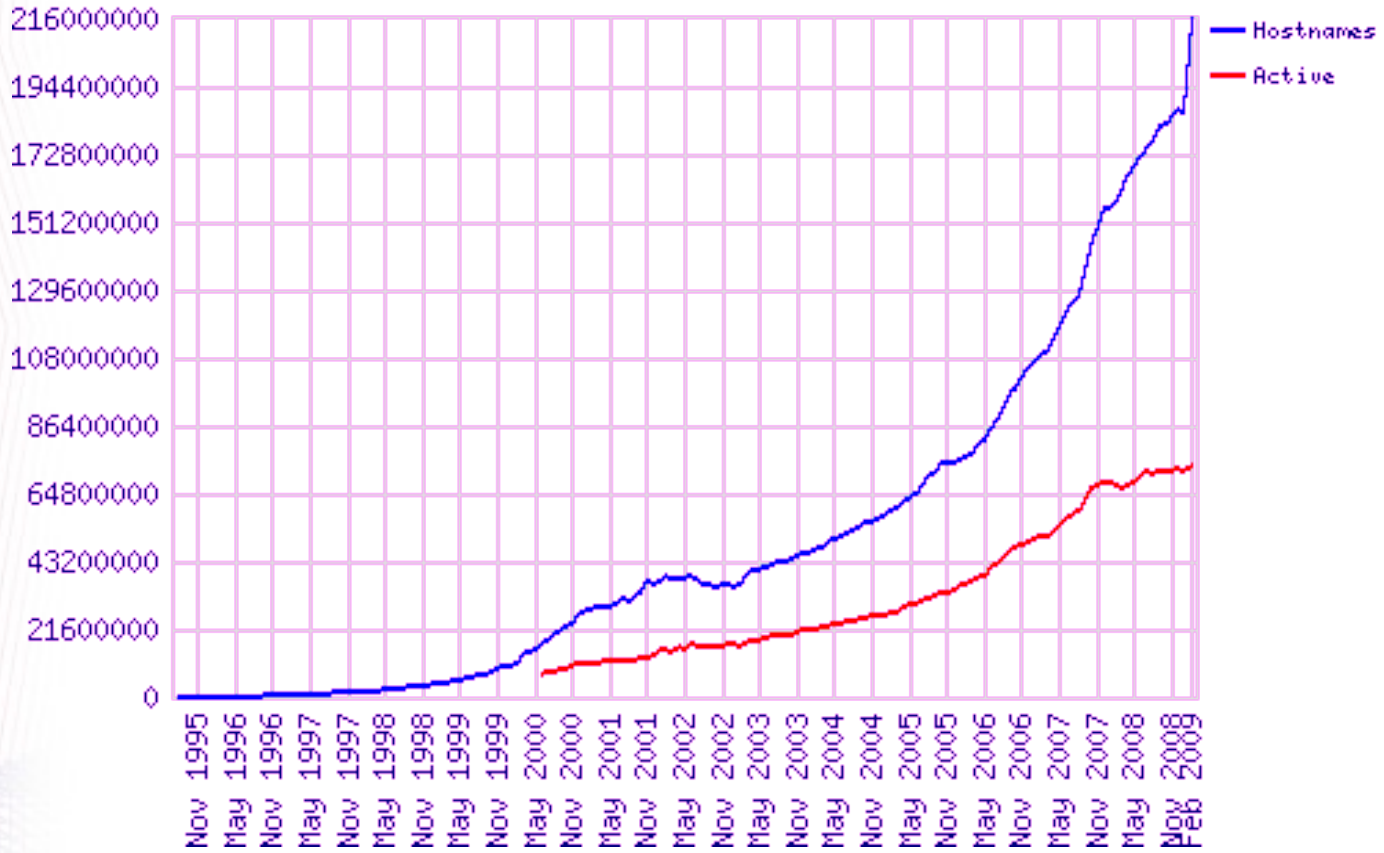
- Weitere Entwicklungen / Portierung auf unterschiedlichste Plattformen
- 1. Dezember 1995: Apache 1.0
- 1999 formierte sich aus der Apache Gruppe die Apache Software Foundation
  - Organisatorische Hilfe
  - Rechtliche Hilfe
  - Finanzielle Hilfe

  Für die Weiterentwicklung des Apache Webserver
- Bis heute wird längst nicht mehr bloss der Apache Webserver unterstützt, die Apache Foundation unterstützt eine Vielzahl unterschiedlicher Open Source Projekten.

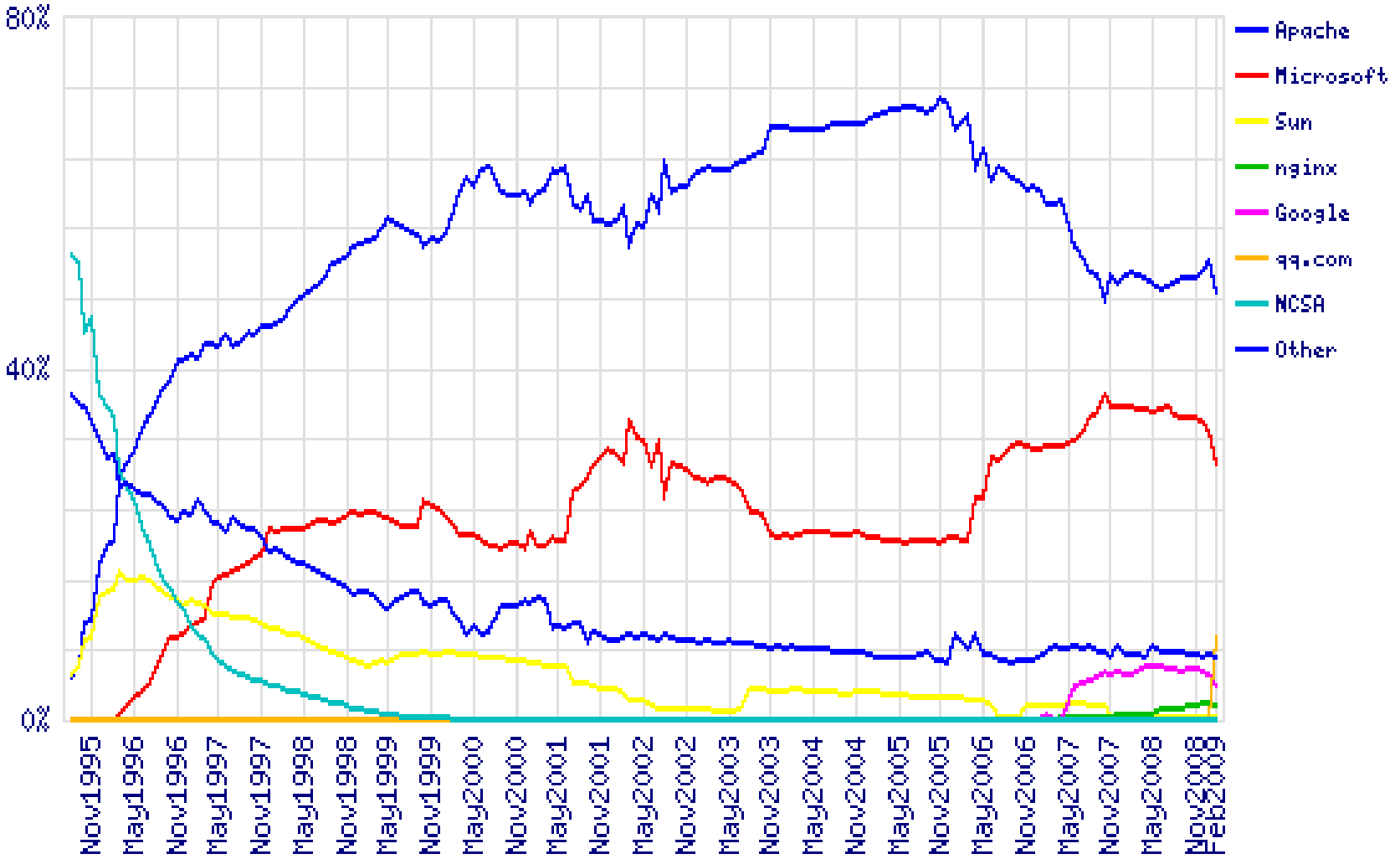
Das Web

- ca. 216 Mio. Hostnamen / 70 Mio. aktiv

(Quelle: http://news.netcraft.com/)



Verbreitung



(Quelle: http://news.netcraft.com/)

Technische Grundlagen

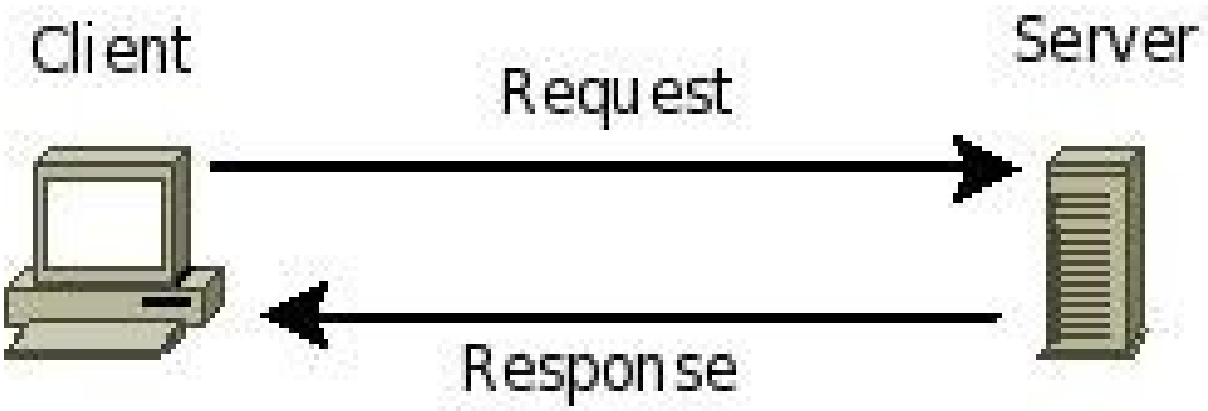
- Serverdienst, welcher im Hintergrund läuft – Loskoppelung vom Elternprozess: Dämon (http**d**)
- Beantworten von Anfragen an Port 80 / 443 (TCP)
  - Dämon wird gestartet
  - Dämon “bindet” sich an den Port 80 / 443
  - Dämon beantwortet Anfragen an diese(n) Port(s)

HTTP

- Hypertext Transport Protocol
- Durch Erweiterungen, Header-Informationen und Statuscodes nicht auf Hypertext beschränkt
- Entwickelt von Tim Berners-Lee im Jahre 1989 am CERN – mitsamt HTML (Hypertext Markup Language) und URL (Unified Resource Locator)
- Das WWW ist geboren
- HTTP wird über TCP abgewickelt, um zuverlässige Übermittlung zu gewährleisten
- HTTP ist ein statusloses Protokoll – es besteht keine dauerhafte Verbindung

HTTP

- HTTP definiert sich durch Frage – Antwort, Request – Response



Request

- Wird durch den Client abgesetzt
- Verschiedene Methoden
  - GET – Anfordern von Inhalten
  - POST – Senden eines Datenblockes
  - HEAD – Nur Kopfeinträge anfragen
  - PUT – Dateien auf den Server hochladen
  - DELETE – Dateien auf dem Server löschen
  - TRACE – Zurückliefern der unveränderten Anfrage
  - OPTIONS – Liste von Server-Fähigkeiten anfordern
  - CONNECT – Für Proxyserver, SSL-Tunnel
- In der Regel werden bloss GET und POST genutzt

Argumente

- Eine Anfrage kann auch Argumente enthalten
  - Per GET
    - Beispiel Google Suche
      - http://www.google.ch/search?hl=de&q=computerlabor&btnG=Google-Suche&meta
        - Entspricht einer GET Anfrage mit den folgenden Parametern:

btnG	Google-Suche
hl	de
meta	
q	computerlabor

Argumente

- Per Post
  - Per Post werden die Daten nicht in der URL sondern im Körper der Anfrage mitgesandt

Frage und Antwort?

- Wie sieht eine Anfrage nun konkret aus? Weshalb weiss der Webserver, ob es sich um eine GET oder eine POST Methode handelt?
  - Request – Response → Frage und Antwort:
    - $ sudo ngrep -d any port 80

GET /apache/ HTTP/1.1..**Host: me.localhost..**User-Agent: Mozilla/5.0 (X11; U; Linux i686; de; rv:1.9.0.6) Gecko/2009030407 Gentoo Firefox/3.0.6 FirePHP/0.2.4..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3..Accept-Encoding: gzip,deflate..Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300..Connection: keep-alive...Pragma: no-cache..Cache-Control: no-cache...

Request also

- Die Methode erscheint als erstes Textelement, gefolgt von der URL (relativ zum Host)
- Host: Weshalb?
  - Pro IP Adresse mehrere Hostnamen möglich (siehe /etc/hosts)
  - DNS liefert IP für Hostnamen
  - → Webserver muss wissen, für welchen Host er die Anfrage beantworten soll

Und die Antwort, Response?

- $ sudo ngrep -d any port 80

T 127.0.0.1:80 -> 127.0.0.1:44350 [AP]

HTTP/1.1 200 OK..Date: Sun, 08 Mar 2009 21:25:19 GMT..Server: Apache..Last-Modified: Tue, 24 Feb 2009 20:32:
43 GMT..ETag: "3a38b19-13d-463b0056d4412"..Accept-Ranges: bytes..Content-Length: 317..Keep-Alive: timeout=15
, max=100..Connection: Keep-Alive..Content-Type: text/html....<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S
trict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">.<html xmlns="http://www.w3.org/1999/xhtml">.
<head> . <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> . <title>Hallo Welt!</titl
e>.</head>.<body>. Hallo Welt! .</body>.</html>.

- Server antwortet mit Protokollversion HTTP/1.1 und Statuscode 200 OK
- Den Kopfeinträgen folgt der Körper, der Quelltext der betrachteten Seite

Statuscodes

- Status-Code “Familien”
  - 1xx Informationen
  - 2xx Erfolgreiche Operation
  - 3xx Umleitung
  - 4xx Client Fehler
  - 5xx Server Fehler

Kopf und Körper alias Head & Body

- Eine Antwort wird in Head und Body unterteilt
- Im Head stehen “Meta” - Informationen zur Übermittlung, als Name – Werte Paare vorhanden
- Im Body ist die eigentliche Antwort zu finden (HTML Quelltext)

Installation

- Sehr einfach:
  - $ sudo aptitude install apache2

Konfiguration

- Unter Linux sind die System-Konfigurationen unter /etc abgelegt, die Apache Konfiguration daher unter /etc/apache2:

```
$ ls -lh /etc/apache2
-rw-r--r-- 1 root root 9.9K 2008-09-19 15:41 apache2.conf
drwxr-xr-x 2 root root 4.0K 2009-03-08 21:36 conf.d
-rw-r--r-- 1 root root  378 2008-09-19 15:41 envvars
-rw-r--r-- 1 root root    0 2009-03-08 21:36 httpd.conf
drwxr-xr-x 2 root root 4.0K 2009-03-08 21:36 mods-available
drwxr-xr-x 2 root root 4.0K 2009-03-08 21:37 mods-enabled
-rw-r--r-- 1 root root  351 2008-09-19 15:41 ports.conf
drwxr-xr-x 2 root root 4.0K 2009-03-08 21:36 sites-available
drwxr-xr-x 2 root root 4.0K 2009-03-08 21:36 sites-enabled
```

/etc/apache2/apache2.conf

- Grundlegende Serverkonfiguration
- Bindet weitere Konfigurationsdateien ein (Include – Direktive)
- Modulkonfigurationen

<IfModule {Name des Moduls}>

{Name der Direktive} {Wert für die Direktive}

</IfModule>

- Bedingte Konfiguration: Bloss, wenn das Modul geladen ist, gelten die Einstellungen
- Includes:

Include /etc/apache2/mods-enabled/*.load

Include /etc/apache2/mods-enabled/*.conf

- Auslagerung von Konfigurationen

Module aktivieren, deaktivieren

- Ubuntu verfügt über zwei Skripte zur Handhabung von Modulen
  - a2enmod: Aktiviert ein Modul
  - a2dismod: Deaktiviert ein Modul
    - Es werden lediglich Symlinks von /etc/apache2/mods-available nach /etc/apache2/mods-enabled gesetzt / gelöscht

Hosts

- Nochmals in /etc/apache2/apache2.conf

```
# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/
```

- Wieder lediglich Symlinks von vorhandenen Dateien unter /etc/apache2/sites-available

Unser erster eigener Host

- $ sudo vim /etc/apache2/sites-available/me.localhost

```
<VirtualHost *:80>
  DocumentRoot /home/me/public_html/apache/public
  ServerName me.localhost
  <Directory /home/me/public_html/apache/public>
    AllowOverride all
    Order allow,deny
    Allow from all
  </Directory>
</VirtualHost>
```

Im Detail

- <VirtualHost *:80>
  - Jeder “Container” startet mit einem öffnenden “Tag”
  - Jeder “Container” endet mit äquivalentem schliessendem “Tag” (ähnlich HTML)
  - * (Wildcard): Dieser Host gilt für alle IP Adressen, welche zu dieser Maschine gehören.
  - 80: Dieser Host wird über den Port 80 bedient.
- DocumentRoot /home/me/public_html/apache/public
  - Definiert, wo die Dateien für diesen Host zu finden sind
- ServerName me.localhost
  - Für welchen Hostnamen dieser virtuelle Host gilt

Im Detail

- <Directory /home/me/public_html/apache/public>
  - Es folgen spezifische Einstellungen für dieses Verzeichnis
- AllowOverride all
  - Bestimmt, welche Direktiven mittels .htaccess Dateien gesteuert, überschrieben werden können
  - Statt all, folgende Möglichkeiten
    - AuthConfig: Authorisationseinstellungen erlaubt
    - FileInfo: Spezifische Dokument-Typen können kontrolliert werden
    - Indexes: Verzeichnis-Indexierung kann eingestellt werden
    - Limit: Zugriff kann beschränkt, eingestellt werden
    - Options: Spezifische Verzeichnis Optionen können gesetzt werden

Hallo Welt

- Hostname muss bekannt sein:
  - $ sudo vim /etc/hosts

    127.0.0.1 me.localhost me
- Verzeichnis muss existieren:
  - $ mkdir -p public_html/apache/public
- Der Webserver braucht was auszuliefern:
  - vim public_html/apache/public/index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
                <title>Hallo Welt!</title>
        </head>
        <body>
                <p>Hallo Welt!</p>
        </body>
</html>
```

Webserver neu laden

- Nach jeder Änderung einer Konfiguration, muss der Webserver diese neu einlesen → Konfiguration wird beim Start eingelesen (oder bei “graceful” restart, auch reload)
  - sudo /etc/init.d/apache2 reload
- Testen wir unsere Webseite:
  - http://me.localhost/

Moment...

- In http://me.localhost/ findet sich jedoch gar nicht der Name der Datei index.html. Weshalb erhalten wir diese dennoch zu Gesicht?
  - $ cat /etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
  DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

- DirectoryIndex teilt dem Apache mit, nach welchen Dateien gesucht werden sollen.

mod_php

- Wie gewohnt, sehr einfach zu installieren:
  - $ sudo aptitude install php5
    - Falls wir PHP auch als “Kommandozeilen” - Version möchten:
      - $ sudo aptitude install php5-cli
      - Damit können wir PHP Skripte via Konsole ausführen
- Automatisch aktiviert:
  - $ ls -lh /etc/apache2/mods-enabled/php*

lrwxrwxrwx 1 root root 27 2009-03-09 11:09 /etc/apache2/mods-enabled/php5.conf -> ../mods-available/php5.conf

lrwxrwxrwx 1 root root 27 2009-03-09 11:09 /etc/apache2/mods-enabled/php5.load -> ../mods-available/php5.load

- Apache neu starten, um das Modul einzubinden
  - $ sudo /etc/init.d/apache2 restart

PHP testen

- $ cd public_html/apache/public
- $ mv index.html hallo.html
- $ vim index.php

  <?php

  phpinfo();
- http://me.localhost/

PHP Konfigurieren

- Die Einstellungen des PHP Moduls für den Apache ist unter
  /etc/php5/apache2/php.ini
  zu finden
- Analog dazu die Einstellungen für die Kommandozeilenversion unter:
  /etc/php5/cli/php.ini

/etc/php5/apache

- short_open_tag

Ermöglicht es, in PHP Dateien (insbesondere interessant für .phtml Dateien), eine Kurzschreibweise zu verwenden: Statt “<?php” reicht ein einfaches “<?”. Sehr interessant: “<?=” entspricht einem “<?php echo”

- memory_limit

Sollte grundsätzlich erhöht werden (viele Pakete funktionieren nicht mit dieser Beschränkung). Ein guter Startwert wäre 32MB

- register_globals

NIEMALS auf On stellen! Diese Option wird spätestens ab PHP 5.3 abgeschafft.

- post_max_size

Sollte erhöht werden, wenn es möglich sein sollte, insgesamt mehr als 8MB zu senden.

- upload_max_filesize

Gibt die maximale Grösse einer einzelnen gesendeten Datei an (ein Bild mit einer Grösse von 4MB würde nicht akzeptiert).

(-> post_max_size sollte grösser sein als upload_max_filesize)

- allow_url_fopen

Eine nette Sache, dies erlaubt es PHP scripts, Dateien übers Web wie lokale Dateien zu öffnen (Bsp: file_get_contents('http://kire.ch/linux/computerlabor.htm');)

Hierbei sollte allerdings beachtet werden, dass dies unter Verwendung von „unsicheren“ scripten zu einem Sicherheitsrisiko werden kann. Grundsätzlich jedoch empfehlenswert.

/etc/php5/apache

- allow_url_include

Ähnlich wie allow_url_fopen, gilt jedoch für Aufrufe von include() und require() (resp. include_once() und require_once()).

Hiervon ist eher abzusehen, das Einbinden einer Datei auf diese Weise sollte nicht über URLs geschehen (erhebliches Sicherheitsrisko).

.htaccess

- Ermöglicht Beeinflussung des Verhaltens des Webservers per Host
- Beispiel Passwortschutz
  - Neue Passwortdatei erstellen:
    - $ mkdir htpasswd
    - $ htpasswd -c .htpasswd me
  - Neuen Benutzer hinzufügen:
    - $ htpasswd .htpasswd her
  - Benutzer wieder entfernen:
    - $ htpasswd -D .htpasswd me
  - Passwortschutz aktivieren:
    - $ vim public_html/apache/public/.htaccess

```
AuthType Basic
AuthName "Melde Dich erst mo an!"
AuthUserFile /home/me/htpasswd/.htpasswd
Require valid-user
```

Quellen

- Wikipedia (http://de.wikipedia.org)
- Netcraft (http://news.netcraft.com)
- Apache.org (http://www.apache.org)