Apache – Workshop

Über dieses Dokument

Dieser Workshop soll eine Hilfe für Webmaster sein, die sich zwangsläufig mit Suchmaschinen und Suchmaschinenoptimierung befassen. Alles hier dargestellte basiert auf persönlichen Erfahrungen, keinesfalls kann und soll eine Garantie für Richtigkeit übernommen werden.

Dieses Dokument wurde verfasst von Jens Ferner, 2F Promoting & Consulting http://www.2-f.biz. Anregungen und Fragen per Email: kontakt@2fpromoting.com.

Inhaltsverzeichnis

Kapitel 1: Einleitung

Der Apache Webserver ist der wohl am häufigsten im Einsatz befindliche Server im Internet. Jeder Webmaster sollte in der Lage sein, den Apache Server zu bedienen, verstehen und zu konfigurieren. Leider aber ist das Wissen über den Apache nicht so weit verbreitet wie es sein sollte – dabei ist es alles andere als schwierig zu erfassen

Ich versuche in diesem Workshop einen kleinen „Crash-Kurs“ zu geben: Angefangen bei http Grundlagen über die Installation des Apache bis zur Konfiguration und Erweiterung sollte alles enthalten sein ☺

Kapitel 2: http – Eine Einführung

Ohne grundlegendes Wissen von http ist es schwierig, Wissen zum Apache zu sammeln und wirklich zu verstehen. Zu http gibt es ganze Bücher, ich kann hier nicht ernsthaft alles vermitteln was zu http gehört – doch ein paar Basics werde ich einzutrichtern versuchen, damit später nicht längere Erklärungen im Rahmen des Apache notwendig sind. Typische Fragen sind immer wieder: Wo ist der Unterschied zwischen einem GET und POST Request – bzw. was ist überhaupt ein REQUEST und „Was ist ein Status-Code und welche gibt es“

Einleitung – das Protokoll

HTTP steht für „Hypertext Transfer Protocol“ und bietet die Grundlage für den Austausch von Daten im Internet. Wer soweit gehen möchte, kann sogar sagen, das HTTP ist das Internet (TCP/IP möge es mir verzeihen). Das Protokoll bietet die notwendigen Voraussetzungen, um die Abfrage von Daten und das Beliefern von Daten zu ermöglichen

REQUEST und RESPONSE – Daten auf Anforderung

Kern des Ganzen ist der REQUEST. Ein Client –das ist der Rechner der zugreift bzw. aufruft- sendet einen Request aus. Dieser Request geht zielgerichtet an einen Server, der dann ggfs. eine Antwort gibt. Diese Antwort wird als RESPONSE bezeichnet

Im Detail: REQUEST

Zuerst beschäftigen wir uns mit dem Request. Einen solchen Request nutzen wir täglich im Internet, wenn folgendes in einen Webbrowser eingetippt wird, handelt es sich bereits um einen Request:

http://www.netz-id.de/

Klingt nicht aufregend, birgt aber eine Menge mehr Hintergrund, als der normale Benutzer auf Anhieb vermuten mag. So erzeugt diese Eingabe in den Browser einen REQUEST der folgenden Art:

GET / HTTP/1.1
Accept : */*
Accept-Language: de-de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE6; WINDOWS)
Host: netz-id.de
Connection: Keep-Alive

Das ist eine Menge, oder? Bedeutet aber alles nichts wildes, es wird hier dem Server nur mitgeteilt, was er braucht um eine ordentliche Antwort zu fabrizieren:

1) Es handelt sich um einen GET Request (dazu gleich mehr) unter Verwendung des HTT-Protokolls Version 1.1
2) Es wird jeder Dateityp akzeptiert (*/*)
3) Sprache des Browsers : Deutsch
4) Es wird eine komprimierte Form akzeptiert
5) Der User-Agent String beinhaltet Daten zum System, Browser etc. – kann meistens frei eingestellt werden im Browser, wenn man weiß wo ☺
6) Auf welchen Host wird zugegriffen
7) Mitteilung dass die TCP Verbindung für weitere Anfragen geöffnet gehalten werden soll

Ich möchte hier nur auf die verschiedenen Client-Requests eingehen, alles andere erklärt sich halbwegs von selber und sollte zur Vertiefung in entsprechenden Büchern nachgelesen werden (Literaturtipps am Ende)

Ein Client kann unterschiedliche Requests senden, dabei ist GET sicherlich der am meisten verwendete, direkt gefolgt von POST. Hier eine kurze Übersicht mit Erklärungen:

GET	Der GET Request ist wohl der Standard: Hiermit wird ein Dokument angefordert (und üblicherweise auch geliefert). Bei einem GET sendet der Client symbolisch gesprochen den Befehl „gib mir das Dokument“ und erwartet die Lieferung
POST	Anders als GET ist POST eher aktiver: Hier werden Daten zum Server gesendet damit sie ort von einem Programm verarbeitet werden können. Typisch ist die Verarbeitung durch ein Skript von Daten die in ein Formular eingegeben wurden
HEAD / TRACE	Mittels HEAD fordert der Client Informationen zu einem Dokument an – ohne das Dokument selber geliefert zu bekommen. Zu diesen Daten gehören eine eventuelle Komprimierung, die Größe des Dokuments, Daten zum Server etc. TRACE ist ein ganz besonderer REQUEST: Hiermit kann nachvollzogen werden, inwiefern die Header durch zwischengeschaltete Proxies verändert wurde und um welche es sich handelt
PUT / DELETE	Mit HTTP Version 1.1 wurden die Requests PUT und DELETE eingeführt – diese ermöglichen es, direkt Dateien auf den Server zu laden bzw. zu löschen – über das HTT-Protokoll
CONNECT	Genutzter REQUEST für eine HTTPS Anfrage
OPTIONS	Hiermit kann der Client anfragen, welche Optionen der Server dem Client an REQUESTS zur Verfügung stellt

Im Detail: RESPONSE

Wenn ein REQUEST an einen Server gesendet wird, antwortet dieser im Normalfall. Ein solcher Response sieht dabei in etwa so aus:

HTTP/1.1 200 OK
Date: Sat, 04 Feb 2004 12:29:02 GMT
Accept-Ranges: bytes
Server: Apache/2.0.48
Content-Length: 1020
Content-Type: text/html

Last-Modified: Fri, 03 Jan 2004 04:34:20 GMT
Client-Date: Sat, 04 Feb 2004 12:29:20 GMT

Das meiste ist wieder selbsterklärend, die erste Zeile aber bedarf einiger Sätze. Geantwortet wird mit http/1.1, sprich die Version 1.1 wird unterstützt. Das folgende „200 OK“ ist wichtig – hierbei handelt es sich um einen „Statuscode“. Der Server sendet auf Anfragen immer einen Statuscode mit. Welche Status Codes es gibt, wurde von w3.org in der RFC2616 festgelegt und kann jederzeit dort nachgelesen werden (http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1.1)

Die Status-Codes

Es ist von Interesse die Status-Codes zu kennen, hier eine Auflistung der möglichen Codes. Die Codes werden in Bereiche eingeteilt, dabei steht jeder Bereich für eine bestimmte Form der Information:

100er Bereich:	Reine Informationen
200er Bereich:	Request erfolgreich ausgeführt
300er Bereich:	Request wurde weitergeleitet
400er Bereich:	Der Request wurde unvollständig bearbeitet
500er Bereich:	Auf dem Server trat ein Fehler auf

Nun im Detail die Status Codes, ich gehe allerdings nur auf die täglich relevanten ein:

200	OK, alles hat ohne Probleme geklappt
201	Es wurde eine neue URL erzeugt
202	Der Request wurde angenommen, aber es ist nichts passiert

203	Die vom Server gesendeten Daten stammen aus einer dritten Quelle, sind also nicht 100%ig zuverlässig
204	„no Content“ – es wurde kein (neuer) Inhalt geliefert
205	Der Client soll den Inhalt (zB eines Formulares) löschen
206	Es wurde nur ein Teil des Dokumentes angefordert
300	Die Anfrage des Clients kann nicht eindeutig beantwortet werden da mehrere Dokumente in Betracht kommen
301	Das vom Client geforderte Dokument ist auf dem Server nicht vorhanden, liegt aber unter einer anderen Adresse die dem Server bekannt ist. Der Client erhält die neue Adresse und soll diese in Zukunft nutzen
302 / 307	Der ehemalige 302, nun 307, sagt dass das verlangte Dokument sich an einer anderen Adresse befindet, möglicherweise aber bald wieder unter der angeforderten zu finden sein wird
303	Das Dokument liegt unter neuer Adresse, die neue Adresse wird mitgeteilt
304	Ein angefordertes Dokument hat sich nicht verändert
305	Das Dokument soll nicht direkt vom Server sondern einem Proxy geliefert werden
400	Die Syntax des Client-Request ist fehlerhaft
401	Eine Authorisierung (Benutzername/Passwort) ist zum Zugriff auf dieses Dokument nötig

402	Dieser Status-Code teilt mit dass zum Zugriff auf das Dokument eine Bezahlung nötig ist – bisher ohne weitere Bedeutung
403	Zugriff auf dieses Dokument ist untersagt
404	Das angeforderte Dokument ist auf dem Server nicht vorhanden
405	Die Art des Requests durch den Client ist auf dem Server nicht erlaubt
408	Timeout
500	Interner Fehler auf dem Server

Kapitel 3: Die Apache Installation

Apache installieren ist kein Zaubertrick (mehr)

Linux

Unter Linux stehen bei Distributionen ohnehin fertige Pakete zur Installation zur Verfügung. Wer möchte kann auch die Quelltexte kopieren und diese selber kompilieren. Das würde hier aber ernsthaft den Rahmen sprengen. Nach der Installation finden sich die Konfigurationsdateien üblicherweise unter /etc/httpd/

Windows

Windows Benutzer finden auf http://httpd.apache.org/download.cgi fertige Installationspakete. Es kann beruhigt die neueste Version genommen werden, etwa 2.0.48. Der „MSI Installer“ ist die richtige Datei → Einfach downloaden und ausführen. Unter Windows wird der Apache als Dienst installiert und es wird ein „Apache Monitor“ installiert, der in der Taskleiste zeigt ob der Dienst gerade läuft oder nicht. Mit einem Klick auf das Bildchen kann der Apache Server-Dienst angehalten oder gestartet werden

Während der Installation wird nachgefragt, wo der Apache installiert werden soll – es kann ein beliebiges Verzeichnis gewählt werden. Dieses Verzeichnis bitte merken, hier liegen die Dateien die später eine Rolle spielen. Unter /conf finden sich die Konfigurationsdateien, im Verzeichnis /htdocs liegt das Standard Verzeichnis des Webservers

Fertig

Nach der Installation im Browser einfach http://localhost/ eingeben – wenn was zu sehen ist hat die Installation funktioniert

Kapitel 4: Apache konfigurieren

Die Installation des Apache ist ein Kinderspiel – doch hier liegt auch nicht der wahre Anspruch. Erst wenn der Apache einmal installiert ist und läuft fängt die Arbeit an – es gilt den Apache zu konfigurieren und den eigenen Wünschen entsprechend anzupassen. Die Konfiguration des Apache läuft immer zentral über die httpd.conf. Unter Linux ist die Datei meistens unter /etc/httpd/ zu finden, unter Windows im Apache Verzeichnis /conf

Die Konfigurationsdatei ist in sich eigentlich gut dokumentiert. Sie ist in 3 Teile –so genannte Sections- aufgeteilt, ich bespreche jeden Teil einzeln

Der Aufbau dieses Kapitels wirkt anfangs vielleicht etwas befremdlich, doch ich habe mir mit der Struktur etwas gedacht. Zuerst werden die 3 Sektionen besprochen – ohne tiefergehende Referenz. Grund: Wer diesen Abschnitt liest, wird wahrscheinlich noch gar nichts von der Apache Konfiguration wissen und erstmal die Basics brauchen. Getrennt davon folgt dann die „Referenz“ in der die Befehle praktisch zum Nachschlagen aufgelistet sind – wobei der erste Teil dann auf die Referenz verweist

Section 1: Global Environment

Hier werden die „wesentlichen“ Informationen abgelegt. Eher selten wird hier eingegriffen, da Änderungen hier den echten Kern des Programms betreffen.

ServerRoot spezifiziert, wo sich die Dateien des Apache befinden. Häufiger Fehler von Anfängern: Es geht hier nicht um die Lage des Roots, also des Ortes an dem sich die Webseiten befinden! Es geht hier um die Lage der wichtigen Dateien des Servers – es muss nur geändert werden, wenn die Dateien woanders liegen als dort angegeben

Wichtige Verbindungsdaten sind Timeout, KeepAlive, MaxKeepAliveRequests und KeepAliveTimeout. Hiermit wird angegeben nach wie vielen Sekunden die Anfrage als Timeout bewertet wird. Ausserdem ob „Keepalive“ aktiviert wird und wie lange Requests „am leben erhalten werden“. Zur Erklärung: Mittels „Keepalive“ ist es möglich, in einer Verbindung zwischen Server und Client mehrere Requests zu senden – so dass nicht jedes Mal eine neue Verbindung eröffnet werden muss. Das ganze ist Ressourcensparender – muss aber individuell eingestellt werden. Die Standardwerte reichen normalerweise.
Bei „***Listen***“ wird angegeben,, auf welchem Port der Apache Server auf eingehende Requests „lauscht“. Es kann nicht nur ein Port angegeben werden, sondern auch eine spezielle IP

Der letzte Teil in Sektion 1 sind die ***Module*** – hier werden Erweiterungen des Apache geladen. Durch die Module werden spezielle Funktionen im Apache überhaupt erst ermöglicht. Passend dazu gibt es noch die Abfrage <IfModule> mit der Befehle nur ausgeführt werden, wenn ein bestimmtes Modul geladen ist.

Zum Laden eines Modules

LoadModule MODULNAME

angegeben. Genaue Informationen zu den Modulen gibt es Kapitel 5.

Section 2: 'Main' server configuration

Die Sektion 1 ist nicht allzu spannend → in der Sektion 2 dagegen geht es bereits ans Eingemachte. Hier werden wesentliche Einstellungen vorgenommen die sich auf alle eingerichteten Webseiten auswirken. Mit diesem Bereich sollte man sich auskennen!

Angegeben wird zuerst der ***ServerAdmin***, das heißt dessen Email-Adresse. Diese wird auf generierten Seiten (typisch ist eine 404 Fehlerseite) angezeigt. Ebenso wird der ***ServerName*** angegeben. Bedeutsam ist ***DocumentRoot***, hier wird angegeben wo sich die das Root-Verzeichnis –das sind die eigentlichen Webseiten- des Webservers befindet

Im Folgenden können per <Directory> verschiedene Verzeichnisse mit Rechten angegeben werden. Dabei wird zuerst angegeben, welches Verzeichnis konfiguriert wird, es folgen dann die Rechte bzw. Optionen. Das Gerüst sieht dabei so aus

<Directory „c:/server“>
Optionen
</Directory>

Als Optionen können verschiedene Werte angegeben werden. Hier die Möglichkeiten:

ErrorDocument

HostNameLookups

IdentityCheck

Options
Require
Satisfy
UseCanonicalName

Weiter geben Sie mittels DirectoryIndex an, welche Dateien als Standard angezeigt werden sollen. Hier fehlt nach der Installation von PHP meistens die index.php die Sie dann nachtragen müssen. Auch fehlt die index.htm in der Standardinstallation

Mittels AccessFileName können Sie einstellen, welche Datei für zusätzliche Konfigurationen genutzt werden können. Standard ist hier die Datei .htaccess, es gibt nur selten einen Grund das ändern zu müssen.

Ähnlich <Directory> arbeitet <Files>. Mit dieser Einstellung können Optionen für einzelne Dateien bzw. Dateitypen festgelegt werden. Standardmäßig ist folgendes eingestellt:

```
<Files ~ "^\.ht">
   Order allow,deny
   Deny from all
</Files>
```

Das untersagt den Zugiff auf jede Datei die mit „.ht“ beginnt

Sehr praktisch ist die Funktion „alias“ mit der Alias-Verzeichnisse angelegt werden können. Das bedeutet, es werden virtuelle Verzeichnisse erstellt, die auf jeder Webseite zur Verfügung stehen und auf ein real existierendes Verzeichnis verweisen. Ein Standard ist hierbei

```
Alias /icons/ "C:/Programme/Apache Group/Apache2/icons/"
```

Das bedeutet es wird in jedem WEB ein Verzeichnis /icons/ (das dort gar nicht existiert!) zur Verfügung stehen und beim Aufruf sieht der Benutzer den Inhalt des „echten“ Verzeichnisses

Der letzte grosse Abschnitt im Bereich „Main“ dreht sich rund um Logfiles. Hier wird eingestellt, was mitgeloggt wird, in welcher Forum und vor allem: Wo das ganze gespeichert wird. Über die Option ***LogLevel*** geben Sie an, welche Meldungen im Error-Logfile gespeichert werden. Möglich sind folgende Werte:

emerg:	Fehlermeldungen beim Start des Servers
alert:	Unerwartete Fehler
crit:	Kritische Meldungen beim Server start
error:	Fehlermeldungen
warn:	Warnungen
notice:	Hinweise
info:	Informationen
debug:	Debug-Meldungen

Diese Liste ist hierarchisch, sprich: Eine Meldungs-Ebene umfasst auch immer Meldungen der darunterliegenden Ebenen. In welcher Datei die Meldungen gespeichert werden geben Sie mittels ***ErrorLog*** an:

```
ErrorLog logs/error.log
```

Üblicherweise werden Sie natürlich mehr erfassen wollen. Dazu gehen Sie in 2 Schritten vor: Per LogFormat definieren Sie zuerst ein LogFormat, dieses erhält einen Namen. Später legen Sie fest, wo Zugriffe in diesem Format gespeichert werden sollen.
Ein Beispiel:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access.log common
```

Die vielen kryptischen Zeichen mit “%” stehen dabei für die jeweiligen Werte, also IP des Benutzers, referer etc. Im Detail dazu weiter unten in der Referenz. Am Ende steht im Beispiel „common“, das ist der Name dieses Formates, das heisst dem Namen „common“ wird dieses spezielle Logfile-Format zugeordnet.

Per CustomLog wird dann ein Logfile (hier logx/access.log) festgelegt, indem die Zugriffe in der Form „common“ abgespeichert werden. Sie können beliebig viele Formate und Logdateien anlegen! Insbesondere bei den Virtualhosts kann dies genutzt werden um für jeden Host eine eigene Logdatei anzulegen

Section 3: Virtual Hosts

Eher selten ist der Fall, dass man einen Webserver einrichtet um dann nur eine Adresse zu bedienen. Vielmehr wird es der Regelfall sein, dass gleich mehrere Präsenzen, unter verschiedenen Adressen „gehostet“ werden sollen. Genau zu diesem Zweck sind die „VirtualHosts“ des Apache Server eingerichtet. Diese ermöglichen es, verschiedene Hosts (www.irgendwas.tld, www.irgendwas2.tld etc.) einzurichten und zu verwalten.

In der Sektion 3 wird mit NameVirtualHost festgelegt, welche Virtuellen Hosts vom Apache Server bedient werden sollen. Standard ist

NameVirtualHost *:80

damit werden dann sämtliche Anfragen über Port 80 abgefangen. Es können auch nur Anfragen einer bestimmten IP oder eines bestimmten Ports abgefangen werden

Um einen virtuellen Server einzurichten genügen ein paar Zeilen:

```
<VirtualHost testjens>
   ServerAdmin webmaster@dummy-host.example.com
   DocumentRoot c:/webserver/jens
</VirtualHost>
```

Nach dem Neustarten des Apache würde unter http://testjens/ das Verzeichnis c:\webserver\jens zur Verfügung stehen – allerdings gibt es ein Problem: Wenn man die Adresse in den Browser eintippt, wird nur ein „Server nicht gefunden“ zu sehen sein. Grund: Ihr Rechner muss erstmal wissen, dass sich hinter „testjens“ die Adresse 127.0.0.1 (localhost) verbirgt. Damit Ihr Rechner das weiss, müssen Sie unter Windows in der „hosts“ Datei einen Eintrag vornehmen.

Sie finden diese Datei entweder unter /Windows/hosts (Windows 98/ME) oder unter /Windows/System32/drivers/etc/hosts (Windows 2000/XP). Dort genügt ein Eintrag

```
127.0.0.1       testjens
```

und schon wird der virtuelle Host gefunden

Im weiteren können Sie die üblichen Optionen angeben (siehe in der Referenz), am bedeutsamsten wird es in der Regel sein, eigene Logfiles für jeden virtuellen Host zu erstellen.

Referenz

Bisher habe ich in fließendem Text versucht eine Einführung zu geben bzw. einen Überblick zu verschaffen. Zum direkten Nachschlagen folgt nun eine Referenz über die einzelnen Befehle die in der Konfigurationsdatei gegeben werden können. Sollte für einen Befehl ein Modul nötig sein, wird darauf hingewiesen

AccessConfig

Konfiguration VirtualHost
Directory .htaccess

AccessFileName

Konfiguration VirtualHost

Damit direkt aus dem Web heraus Einstellungen vorgenommen werden können, kann hier eine Datei vorgegeben werden die zusätzlich eingelesen wird. Standard ist .htaccess

AccessFileName .htaccess

AllowOverride

Directory

Soll es möglich sein zusätzliche Einstellungen über eine per AccessFileName definierte Datei vorzunehmen? Wenn ja: Welche Einstellungen sind möglich:

All	-	Alles ist möglich
AuthConfig	-	Sämtliche Optionen rund um Auth sind möglich
FileInfo	-	Alles rund um Dokumente kann eingestellt werden
Indexes	-	Die Indexes-Optionen können vorgenommen werden
Limit	-	Betrifft allow, deny, order
None	-	Eine eventuell vorhandene Datei wird ignoriert
Options	-	„Options“ sind möglich

AllowOverride ALL

*Auth**

Directory .htaccess

Es gibt einige Auth-Optionen, hier alle in einer Übersicht:

AuthGroupFile* :	Bezeichnung einer Datei die Benutzergruppen mit Zugriff enthält
AuthName :	Name des geschützten Bereichs
AuthType :	Art der Authentifikation (Basic / Digest), Basic ist üblich
AuthUserFile* :	Name einer Datei die Benutzer und Passwörter mit Zugriff enthält

*): Modul mod_auth ist nötig

<Directory>

Konfiguration VirtualHost

Es werden Rechte und Optionen für einzelne Verzeichnisse gesetzt

<Directory>

Konfiguration VirtualHost

Es werden Rechte und Optionen für einzelne Verzeichnisse gesetzt

Wird später fertig gestellt!

Apache auf der Kommandozeile

Sie können den Apache natürlich auch auf der Kommandozeile starten – jeweils mit unterschiedlichen Optionen

Wird später fertiggestellt

Kapitel 5: Apache erweitern

PHP installieren

Apache installieren ist kein Problem: Einfach die Setup-Datei ausführen und der Rest läuft von alleine. Bei PHP sieht das aber anders aus: Auf PHP.net findet man zwar eine Setup-Datei, die ist aber nicht in der Lage sich mit dem installierten Apache Webserver automatisch zu konfigurieren. Problem? Nein – es ist etwas ungewohnt für den „normalen“ Windows® Benutzer, aber keinesfalls unmöglich

Sie besorgen sich zuerst die „Windows Binaries“ im ZIP File auf http://www.php.net/downloads.php. Das ZIP Paket entpacken Sie wo Sie es am liebsten haben möchten. Auf der obersten Ebene der entpackten Datei finden Sie eine php4ts.dll. Diese Datei kopieren Sie in das Verzeichnis c:\windows\system32 – das ist von enormer Bedeutung! Ausserdem finden Sie auf der obersten Ebene eine Datei php-ini.dist; Diese Datei benennen Sie um in php.ini und kopieren Sie in das Verzeichnis c:\windows. Dies ist die Konfigurationsdatei von PHP, ich gehe hier nicht auf die verschiedenen Optionen ein sondern schreibe dazu einen eigenen Workshop. Suchen Sie auch auf Netz-ID.de einmal dazu wenn Sie Fragen haben oder nutzen Sie das Forum auf Nukeboards.de

Damit PHP auch im Apache läuft müssen Sie in der httpd.conf noch das Modul laden. Dazu fügen Sie in die Konfigurationsdatei folgende Zeilen ein:

```
LoadModule php4_module c:/Webserver/php/sapi/php4apache2.dll
AddType application/x-httpd-php .php
```

Dabei müssen Sie natürlich Ihr PHP-Verzeichnis richtig angeben, bei mir ist das c:/Webserver/php/, kann natürlich dann variieren

Das wars schon, einfach den Apache neu starten und schon sollte PHP in den Apache eingebunden sein. Sollte der Apache nicht neu starten, einfach einmal die Dateien aus /dlls in das Windows/system32 Verzeichnis kopieren – eventuell hilft das schon!

mySQL installieren

Die mySQL Datenbank zu installieren ist so wie beim Apache kein Problem: Einfach das Archiv auf http://www.mysql.com/downloads/ kopieren und entpacken. Darin befindet sich eine Setup-Datei, die muss ausgeführt werden. Führt die Setup-Datei zu einem seltsamen Fehler, das Verzeichnis in dem die Setup Datei liegt einfach umbenennen in etwas kürzeres, zB mysqlinst. Die Setup Routine scheint Probleme mit komplizierten Namen des Verzeichnisses zu haben in dem sie liegt

Nachdem mySQL einmal installiert ist findet sich im Installations-Verzeichnis ein Unterverzeichnis /bin. Auf der Eingabeaufforderung in dieses Verzeichnis wechseln und

```
Mysqld-nt –install
```

eingeben. Danach wird mySQL als Dienst eingerichtet (Windows XP, 2000) und wird automatisch, bei Bedarf gestartet

Den Apache testen mit „ab“

Zum Apache gehört ein schönes Tool – es nennt sich apachebenchmark „ab“ und dient dazu, Performance-Tests auszuführen. Dazu werden beispielsweise eine bestimmte Anzahlzugriffe pro angegebenen Zeitraum auf den Server ausgeführt und man kann den Datendurchsatz und die Verlustrate testen

Ein einfaches Aufrufen von „ab“ ohne Angabe von Argumenten gibt alle möglichen Optionen aus. Der typische Test wird dieser sein

ab –n 500 –c 5 http://localhost/test/index.html

Das heißt es werden 500 Zugriffe, dabei immer 5 gleichzeitig auf die angegebene Seite ausgeführt. Zurückgegeben wird eine detaillierte Statistik

Die Module

An dieser Stelle folgen später Hinweise zu den Modulen des Apache

Kapitel 6: Typische Probleme

Dieses Kapitel folgt später!

Schutz mit Benutzername & Passwort

Sicherheit im Allgemeinen