



Schneller liefern, weniger Last

Server Caching



Viele Sites leiden nur deshalb unter zu **hoher Last**, weil sie die zur Verfügung stehenden Mechanismen nicht gut genug verwenden. **Der wichtigste** davon ist **das Caching**.

THOMAS WÖLFER

Sie können alle möglichen Dinge cachen, und was Sie cachen ist auch dafür verantwortlich, wie Sie das am besten tun müssen. Den „Caching-Ein“-Schalter, der Sie rundrum prima versorgt, den gibt es leider nicht. Daher beschäftigt sich dieser Artikel auch mit verschiedenen Aspekten beim Caching. Diese sind nur teilweise über das Programmieren zu lösen.

Dynamische Webseiten cachen

Über kurz oder lang entdeckt jeder Web-Master dynamische Webseiten, ganz gleich ob per ASP, PHP oder sonst einem Mechanismus. Dynamische Webseiten sind ungeheuer praktisch, wenn es darum geht, Informationen mit

ne dynamische Webseite zu gestalten. Dafür nimmt man zwei bestimmte Informationen aus einer Datenbank und baut daraus die eigentlich zu liefernde

Das ist bei wenigen Besuchern auf der Webseite kein Problem, auch wenn die geschilderten Vorgänge dramatisch aufwändiger sind, als wenn der Apache ein-

```
ssh-nickles - PuTTY
F10 key => File Edit Search Buffers Windows System Help
$C = mysql_connect( SITE_DB_SERVER, SITE_R_USERNAME, SITE_R_PASSWORD);
$r = mysql_db_query( "nickles_msgs", "select * from daily_news where id=" . $
$o = mysql_fetch_object( $r);

// begin building the html
$documentBody = new DocumentBody;
// ako entscheidet ob js eingebunden wird oder nicht
// wichtig muß unter dem <body> stehen
EmitMenuSelection();

$documentBody->EmitBannerCode( $o->category);

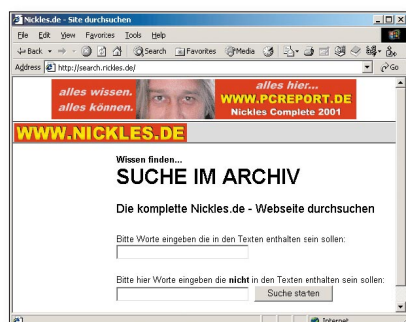
//CreateNavBar( 0, "http://www.nickles.de/", "http://www.nickles.de/", "", "$
// ako hat einen default-parameter der auf 100 (Pixel von oben) gesetzt ist
CreateNavBarDynamicMenu();

print '<table width=100% border=0 cellpadding="0" cellspacing="0"><tr>';

$documentBody->EmitVertikalBlockSpacer();

// prepare left margin and emit it
C+(Jed B0 99 9) Emacs: build news.php3 (Text) 32/122 6:43pm
```

EINE TYPISCHES PHP-SCRIPT, das eine Webseite zusammenbaut: Solche Seiten sollten Sie cachen.



EINIGE SEITEN wie zum Beispiel Suchseiten lassen sich nicht ohne weiteres cachen, da die Suche meist vom Surfer abhängig ist.

unterschiedlicher Haltbarkeit zusammenzufassen und zu veröffentlichen. So ist es zum Beispiel möglich, in PHP ei-

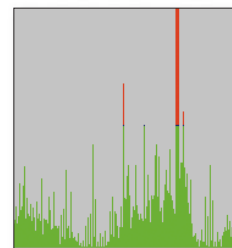
Seite auf. Eine solche PHP-Seite könnte in etwa folgendem Aufbau haben: siehe Listing 1 im Kasten auf der übernächsten Seite.

Jedesmal wenn diese Seite von einem Client angefordert wird, passiert Folgendes: Zunächst wird der PHP-Prozessor vom Apache angestellt. Dieser arbeitet das Script ab und erzeugt dabei HTML-Code. Dieser Code wird dem Client geliefert.

Im Zuge des Abarbeitens werden nicht nur verschiedene Funktionen aufgerufen (die zum Teil sogar zu Aufrufen des Betriebssystems führen können, was ein „teurer“ Vorgang ist), sondern es wird auch zweimal der MySQL Datenbank-Server abgefragt. Auch dieser Vorgang erzeugt ein nicht unerhebliches Maß an Last.

fach eine auf der Festplatte vorliegende Datei als Datenstrom direkt zurückliefern könnte.

Wenn Ihre Webseite allerdings viele Besucher hat, machen sich diese Vorgänge sehr schnell bemerkbar: Der Ser-



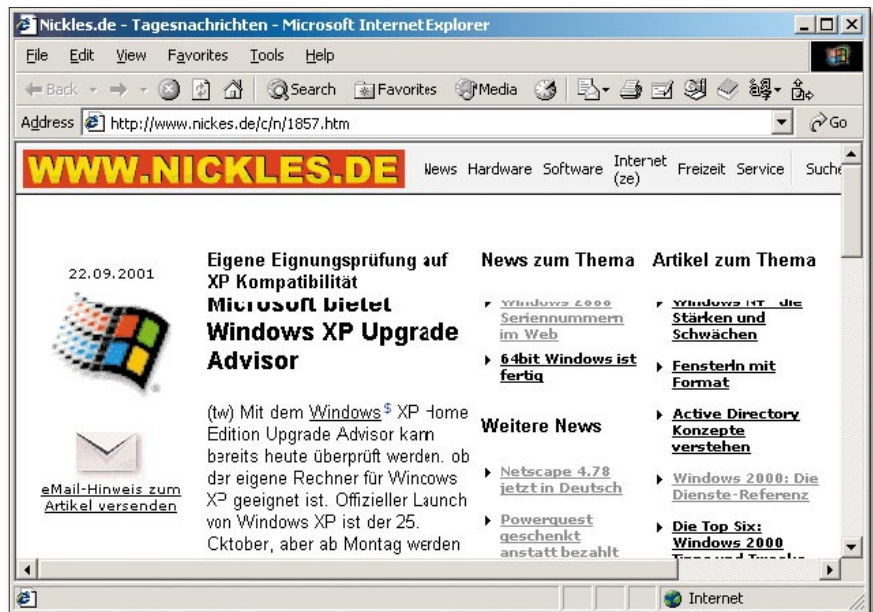
SERVER-ÜBERLAST entsteht leicht, wenn Sie zu viele dynamische Seiten auf Ihrem Server verwenden.

ver wird in kürzester Zeit überlastet sein, zumindest dann, wenn alle Ihre Seiten auf diese Art erzeugt werden. Wann die Überlast des Servers erreicht ist, hängt

von zwei Dingen ab: von der Anzahl und Komplexität Ihrer PHP-Programme und von der Anzahl Ihrer Besucher.

Dieses Problem ist nicht neu und wird bei größeren Sites normalerweise durch den Einsatz eines Staging-Servers und eines Content-Management-Systems gelöst. Dabei werden die Seiten anhand von Daten, die das Content-Management-System verwaltet, zu festgesetzten Zeiten dynamisch generiert. Das Produkt davon wird in Form statischer Seiten auf dem Staging-Server gespeichert, und diese statischen Seiten werden dann ihrerseits zu festgelegten Zeitpunkten auf den „Live“-Server kopiert. Der „Live“-Server ist derjenige, den die Surfer zu Gesicht bekommen. Das bedeutet, es werden weiterhin Seiten dynamisch zusammengesetzt – aber die Surfer sehen nur statische Abbildungen davon. So muss nicht jede Seite für jeden Besucher neu erzeugt werden – das senkt die Last auf dem Server ganz erheblich.

Dieser Vorgang hat aber auch einen Haken: Content-Management-Systeme sind im Allgemeinen nicht nur recht teuer, sondern auch nicht immer so flexibel



EINE TYPISCHE NEWS-SEITE: Nachdem sich der Inhalt der News kaum kurzfristig ändert, lohnt es sich solche Seiten zu cachen.

wie man sich das wünschen würde. Außerdem verlangt der Einsatz eines solchen Systems fast immer einen zweiten Server – für kleinere Websites oder

Budgets verbietet sich der Einsatz solcher Systeme daher fast von selbst. Es gibt aber auch eine einfachere, deutlich preiswertere Lösung für das geschilder-

CACHE-CONTROL RICHTIG EINGESETZT

Wie zu Beginn des Beitrags erwähnt, leiden Sie als Web-Master immer unter zu wenigen Ressourcen. Von einer Resource haben Sie allerdings immer genug: Surfer und deren Browser.

Das können Sie sich zu Nutze machen. Der Trick hierbei ist der richtige Einsatz von Cache-Controls.

Beim Cache-Control geht es nicht darum irgendwelche Elemente auf Ihrem Server möglichst effizient zwischenspeichern, sondern darum, dass diese Elemente erst gar nicht von Ihrem Server angefordert werden. Normalerweise lädt ein Surfer, der Ihre Seite besucht, alle Elemente dieser Seite – also das reine HTML, die Bilder, JavaScript-Dateien, etc. – herunter. Wenn der gleiche Surfer Sie in wenigen Minuten, Stunden oder Tagen wieder besucht, passiert das Gleiche.

Dabei werden sich einige Seiten in der Zwischenzeit nicht geändert haben. Das Gleiche gilt für praktisch alle Bilder, Logos und dergleichen. Auch JavaScript-Programme, die Sie vielleicht für Ihre Navigation verwenden, werden sich kaum geändert haben. Außerdem hat der Surfer mit recht großer Wahrscheinlichkeit diese Da-

teien bereits einmal heruntergeladen. Alles, was Sie also brauchen, ist ein Mechanismus, um diesen Umstand dem Browser des Surfers bekannt zu machen. Die Daten müssen nicht neu heruntergeladen werden, der Surfer sieht die Seite schneller, und Sie haben weniger Last und Bandbreitenverbrauch. Es ist allgemein bekannt, dass der Browser einen Cache besitzt und Dateien, sofern diese schon im Cache vorliegen, nicht erneut vom Browser heruntergeladen werden.

In der Praxis sieht es jedoch so aus, dass jede Datei, die im Browser-Cache liegt, ein Haltbarkeitsdatum hat. Wird dieses überschritten, fordert der Server die Datei neu an. Die „Default“ Haltbarkeit Ihrer Elemente liegt jedoch bei Null. Das bedeutet effektiv, dass der Browser-Cache gar nicht richtig zum Einsatz kommt, wenn Sie sich auf Ihrem Server nicht darum kümmern.

Sie stellen diese Haltbarkeit im Apache Config-File ein. (Beim IIS können Sie die Haltbarkeit von Objekten im GUI des IIS Dienste-Managers einstellen).

Dazu benötigen Sie das „Expires“ Statement. Sie müssen zum einen den „Ex-

pires“-Mechanismus einschalten – auch das geht per Config-Statement. Ferner müssen Sie die Haltbarkeit der Objekte einstellen. Das geht etwa mit Directory oder File-Containern.

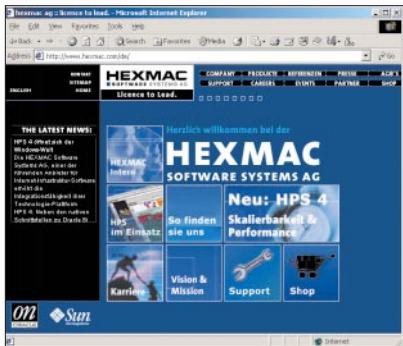
Um zum Beispiel einen kompletten Server mit Haltbarkeitsinformationen zu versorgen, brauchen Sie nur einige wenige Zeilen in Ihrem `httpd.conf`. Angenommen alle Bilder sollen eine Haltbarkeit von 30 Tagen, alle anderen Dateien eine von einem Tag haben. Dann könnten Sie folgenden Eintrag in Ihrer Konfigurationsdatei verwenden:

```
<Directory ./usr/home/Some
➤Host/htdocs">
    ExpiresActive on
    ExpiresByType image/gif
➤"Access plus 30 days"
    ExpiresByType image/jpg
➤"Access plus 30 days"
    ExpiresByType image/png
➤"Access plus 30 days"
    ExpiresDefault "Access
➤plus 1 days"
</Directory>
```

Weiter Informationen über den Expires-Header finden Sie auf www.apache.org



te Problem, die Sie im Folgenden erfahren werden.



CONTENT-MANAGEMENT SYSTEME wie HPS von Hexmac sind zwar praktisch, aber auch meist nicht ganz billig.

Content-Management-Systeme lösen eine ganze Reihe von Aufgaben, die über das reine Last-Problem hinausgehen. Wenn es aber nur um die Senkung der Server-Last und die dadurch entstehende bessere Ausnutzung des Servers geht, ist das das zentrale Problem der Besucher, weil für jeden jede angeforderte Seite neu erzeugt werden muss.

Allerdings haben die meisten dynamischen Seiten gar keinen dynamischen Inhalt – zumindest keinen, der tatsächlich für jeden Besucher neu aufgearbeitet werden müsste. Eine typische News-Seite, wie die oben abgebildete, wird sich eher einmal am Tag ändern. Im Zeitraum dazwischen könnten die Seite eigentlich immer in der gleichen Form verwendet werden. Gleiches gilt für die Startseite von fast jedem Web-Auftritt. Diese wird sich nur hin und wieder ein wenig verändern und nicht bei jedem Abruf.

Daraus ergibt sich eine einfache Lösung des Last-Problems, das praktischerweise keine oder nur sehr wenige Änderungen an Ihrem PHP-Code benötigt, zumindest an dem Code, der die eigentliche Seite zusammenbaut.

Der Trick dabei ist, sich verschiedene Fähigkeiten des Apache zu nutze zu machen. Dabei müssen Sie folgendermaßen vorgehen: Zunächst ändern Sie die Links zu den bisher dynamischen Seiten. Die neuen Links lassen Sie dabei auf ein gemeinsames Verzeichnis zeigen und darin auf normale HTML-Dateien. Um beim News-Beispiel zu bleiben, könnten Sie etwa statt der alten URL:

www.SomeHost.com/displaynews.php?news_id=123
die neue URL
www.SomeHost.com/news/123.html

verwenden. Mit „123“ ist dabei die Nummer der News gemeint. Das kann zum Beispiel die Nummer der News in Ihrer Datenbank sein, aber auch sonst jeder eindeutige Bezeichner ist o.k.

Die Datei 123.html selbst legen Sie zunächst gar nicht an. Was daraus resultiert ist klar: Sie haben nun lauter Links auf News-HTML Seiten, die gar nicht existieren.

Das hat den Effekt, dass der Apache bei einer Anforderung einer dieser Seiten die „404“ Fehlermeldungs-Seite anzeigen wird. Dazu müssen Sie wissen, dass es beim Apache möglich ist auf Verzeichnis-Ebene anzugeben, welche 404er Seite verwendet werden soll. Um genau zu sein: Sie können sogar ein Programm oder ein PHP-Script installieren, das vom Apache im Falle eines 404er

LISTING 1

```
<?php
// hilfsmethoden
function RenderNewsItem( $o)
{
    // z.b.:
    print $o->strInhalt;
}
function RenderWerbung( $o)
{
    print "<a href=$o->url><img src=$o->img_url></a>";
}
function GetNewsByID( $d)
{
    $c = mysql_connect( "DB_SERVER", "USERNAME", "PASSWORD");
    $qs = "select strInhalt from news where id=$d limit 1";
    $r = mysql_db_query( "NameDerDatenbank", $qs, $c);
    $o = mysql_fetch_object( $r);
    return $o;
}
function GetWerbung()
{
    $c = mysql_connect( "DB_SERVER", "USERNAME", "PASSWORD");
    $qs = "select url,img_url from ads order by rand() limit 1";
    $r = mysql_db_query( "NameDerDatenbank", $qs, $c);
    $o = mysql_fetch_object( $r);
    return $o;
}
// ----- hier geht die eigentliche seite los
print "<html><head><title>Beispiel-Seite</title></head>";
print "<body>";

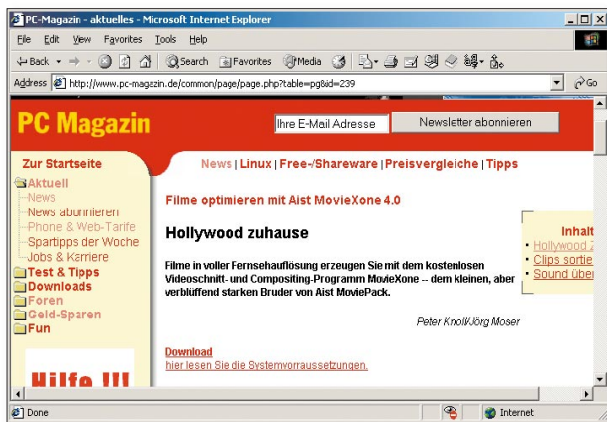
// $id wird beim aufruf der seite uebergeben
$id = GetNewsByID( $id);
$w = GetWerbung();

RenderWerbung( $w);
RenderNewsItem( $id);

print "</body></html>";
?>
```

LISTING 2

```
<?php
header("HTTP/1.0 200 OK");
$pathParts = explode("/", $REQUEST_URI);
$nLastPart = count($pathParts);
$szFilename = $pathParts[$nLastPart - 1];
$szFilenameparts = explode(".", $szFilename);
$szID = $szFilenameparts[0];
if( ! is_numeric( $szID))
{
    print "<html><head></head><body>sorry, die angeforderte url ist
    nicht auf diesem server gueltig</body></html>";
    return;
}
$request_news = "http://www.SomeHost.com/displaynews.php?news_id=$szID";
$news_filename = "/usr/home/somehost/htdocs/news/$szID.html";
$content = join('', file( $request_news));
$file = fopen( $news_filename, "w");
fputs( $file, $content);
fclose( $file);
readfile( $news_filename);
?>
```

PHP-SEITEN SIND zwar praktisch, machen aber eine Menge Last: Für jeden Surfer muss die Seite eigens angelegt werden.

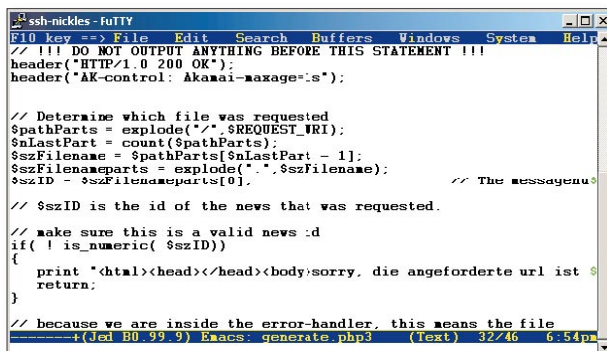
Fehlers aufgerufen wird. Das geht einfach über eine `.htaccess`-Datei die Sie im Beispielfall im Verzeichnis www.SomeHost.com/news platzieren. In dieser Datei können Sie eine ganze Menge an Optionen festlegen, aber für die Definition eines 404er Error-Handlers reicht eine einzige Zeile:

```
ErrorDocument 404 generate.php3
```



MIT EINER .HTACCESS DATEI legen Sie beim Apache fest, welches Script im Falle eines 404er Fehlers ausgeführt werden soll.

Mit `ErrorDocument 404` geben Sie an, dass diese Zeile den Error-Handler für 404er Fehler definiert. „generate.php“ ist einfach nur der Name eines Scriptes, das vom Apache gestartet wird. Sie können hier auch einen vollen Pfad mit angeben oder einen anderen Namen verwenden. Wenn Sie die abgebildete Zeile einfach übernehmen, erwartet der Apache den



IM ERROR-HANDLER können Sie natürlich beliebige PHP-Befehle verwenden. Wichtig ist, dass Sie kein HTML ausgeben, bevor Sie die `header()`-Funktion verwenden.

Error-Handler unter dem Namen „generate.php“ in dem Verzeichnis, in dem auch der Fehler aufgetreten ist.

Das „generate.php“-Script kann nun alles tun, was normale PHP-Skripte auch können. Sie könnten zum Beispiel einfach die normale Seite zusammenbauen und zurückliefern. Das wäre aber nicht sonderlich sinnvoll, denn dadurch wäre ja

nichts gewonnen.

Statt dessen schreiben Sie ein Script, das mehrere Aufgaben ausführt:

- Zunächst liefern Sie einen passenden

http Header – und zwar einen, der dem Client eben keinen 404er Fehler signalisiert, sondern ein ganz normales OK mit „200“.

- Dann untersuchen Sie die angefragte URL, und ermitteln welche News-Seite eigentlich angefordert wurde. Wichtig ist dabei, dass Sie wie zuvor beschrieben einen eindeutigen Identifizierer in der URL angebracht haben.

- Nun bauen Sie eine statische Kopie der angeforderten Seite zusammen. Das geht zum Beispiel dadurch, dass Sie Ihr ursprüngliches Script verwenden.

- Schließlich liefern Sie diese Seite zurück.

Ein solches Script würde beim geschilderten Beispiel etwa das folgende Aussehen haben: siehe Listing 2 im Kasten auf der vorherigen Seite.

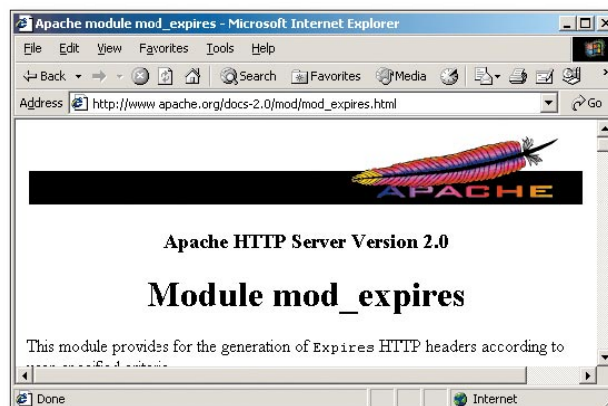
Das abgebildete Script passt zum bisher beschriebenen Beispiel. Zunächst wird mit der `header()`-Funktion ein gültiger HTTP-Header emi-

tiert. Ausgegeben wird dabei einfach ein `OK`, der Browser erkennt dadurch, dass er die Seite in funktionstüchtigem Zustand erhalten hat. Bei der Header-Funktion ist dies wichtig, da Sie vorher keinerlei HTML ausgeben, ansonsten schlägt sie fehl.

Nun ermitteln Sie die „Nummer“ der gewünschten News. Diese ist als Teil der angeforderten URL vorhanden. Die URL hat den Aufbau `http://www.SomeHost.cpm/neww/123.html`.

Zunächst verwenden Sie die `explode()`-Funktion, um die URL in Ruhe durch einen „/“ in getrennte Teile zu zerlegen. Dann zählen Sie die Anzahl der Teile. Das ist allgemeingültiger, als einfach einen festen Index zu verwenden. Somit können Sie das Script auch für andere Verzeichnisse verwenden.

Der letzte Teil der URL `$pathParts[$nLastPath - 1]` enthält den reinen Da-



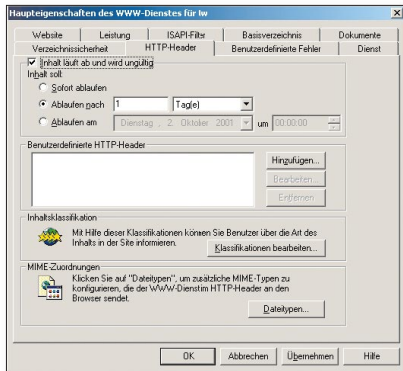
MIT DEM MOD_EXPIRES können Sie beim Apache die Haltbarkeit Ihrer Elemente festlegen.

teinenamen, als `123.html`. Diesen String zerlegen Sie nochmals mit `explode()`, diesmal benutzen Sie aber einen „.“ als Separator. Das liefert Ihnen in `$zFile-NameParts[0]` die gesuchte Nummer. Zumindest sollte diese an dieser Stellen stehen.

Nun kann aber ein Surfer auch irgend etwas völlig willkürliches eingegeben haben – also eine Anforderung an eine Seite die weder im `/news`-Verzeichnis liegt, und auch von Ihrem Script nicht erzeugt werden kann. Diese Fehlermöglichkeit behandeln Sie im nächsten Schritt. Da Sie ja wissen, dass es sich beim bisher ermittelten String um eine Nummer handeln muss, gelingt das mit dem Statement

```
if ( ! is_numeric( $szID ) )
```

Hier wären natürlich noch weitere Tests denkbar – zum Beispiel könnten Sie überprüfen, ob die Nummer eine für Ih-



BEIM IIS KÖNNEN Sie die Haltbarkeit von Elementen mit dem IIS Snap-in festlegen.

re News gültige Nummer ist oder nicht. Tritt eine Fehlerbedingung ein, liefern Sie einfach eine entsprechende Meldung und verlassen Ihr Script per *return*.

Im folgenden Teil des Scriptes bauen Sie die statische Seite zusammen. Sie beginnen zunächst damit, die URL zum „alten“ News-Script zusammenzubauen, und zwar mit dem benötigten ID-Parameter für die News:

```
$request_news = "http://www.
SomeHost.com/displaynews.php3?
news_id=$szID";
```

Außerdem benötigen Sie einen String, der den kompletten Namen der eigentlich benötigten Datei enthält. Dieser setzt sich einfach aus dem lokalen Pfad auf das News-Verzeichnis und der passenden Nummer zusammen, also zum Beispiel:

```
$news_filename = "/usr/home/so
mehost/htdocs/news/$szID.html";
```

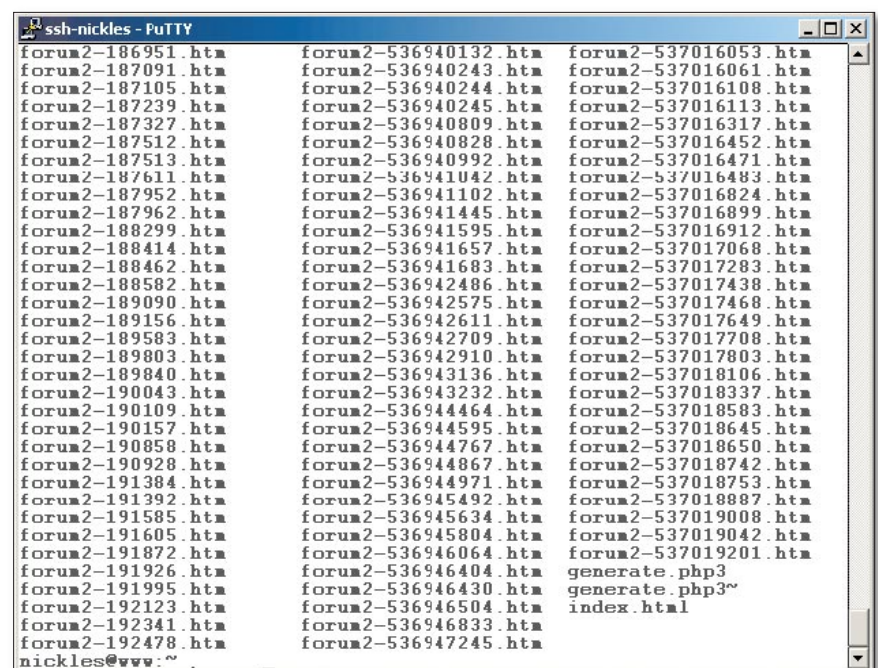
Nun lassen Sie selbst einen Request auf die URL des alten Scriptes los. Dieses baut die News-Seite komplett zusammen. Die dadurch gelie-

ferte Seite speichern Sie in einer Variablen:

```
$fcontents = join('', file( $re
quest_news));
```

Schließlich öffnen Sie die „neue“ statische Datei und geben den zuvor erfragten Inhalt aus dem Script in die Datei aus. Dann schließen Sie die neue statische Datei. Diese ist nun fertig und enthält alle benötigten Daten:

```
$file = fopen( $news_filename,
"w");
fputs( $file, $fcontents);
fclose( $file);
```



NATÜRLICH MÜSSEN SIE irgendwann einmal die erzeugten Seiten löschen. Den Zeitpunkt legen Sie am besten mit einem Cron-Job fest.

Zuletzt liefern Sie die so zusammengesetzte Datei an den anfragenden Client zurück:

```
readfile( $news_
filename);
```

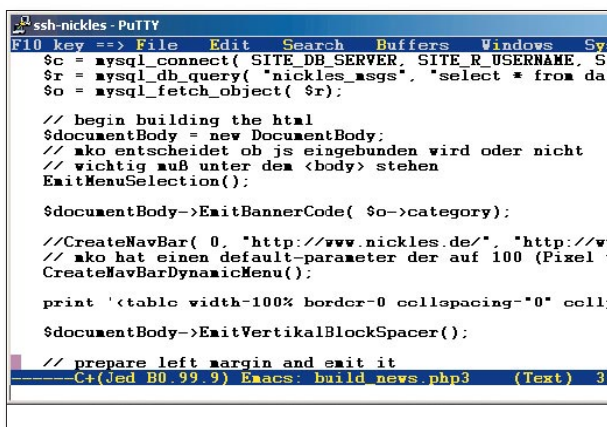
Es wird dann folgender Vorgang ausgelöst. Beim ersten Request einer bestimmten News-Seite existiert diese Seite noch nicht. Daraufhin startet der Apache Ihres Error-Handler.

Dieser wiederum kümmert sich darum, dass die Seite dynamisch generiert und später dann als statische Variante gespeichert wird.

Diese Variante wird dann zurückgeliefert.

Beim nächsten Request auf diese Seite existiert sie schon. Das bedeutet, dass der komplette Schritt, die Seite „dynamisch“ zu erzeugen, entfallen kann. Der Apache wird sie einfach ausliefern, ohne dass weitere Schritte notwendig werden.

Sie müssen sich nur noch überlegen, in welchen Zeitabständen Sie Ihre News-Seiten neu aufbauen möchten. Unter Umständen müssen einmal erzeugte Seiten nie wieder neu gebaut wer-



AUF DER WEB-ADRESSE www.php.net finden Sie nicht nur PHP selbst, sondern auch die komplette Dokumentation zu dieser Sprache.

den – andererseits können diese Seite auch Elemente enthalten, die Sie auf Tagesbasis oder in anderen Perioden verändern müssen. Das lösen Sie einen Cron-Job der alle *.html Seiten aus den /news-Verzeichnis in regelmäßigen Abständen löscht. Fertig ist das PHP-Caching.

In diesem Beitrag haben Sie erfahren, wie Sie Ihre dynamischen Webseiten auf preiswerte und vor allem einfache Art und Weise in statische Seiten umbauen können, ohne den Vorteil der dynamischen Komponenten zu verlieren. Auf diese Weise sollten Sie eine Vielzahl von weiteren Surfern gleichzeitig auf Ihrem Rechner verwalten können. Praktischerweise geht das, ohne dass Sie Ihre bisherigen Scripts verändern müssen. Viel Spaß beim Ausbau Ihres Web-Angebotes. 