Web-Server: Apache-Optimierung auf HTTP-Ebene

Wieso auf HTTP-Ebene?

Mit den grundsätzlichen Apache-Optimierungen ist der kleine Admin häufig bereits überfordert oder hat zuwenig Geduld und Erfahrung.

Daher hier ein paar Tricks und Kniffe um den Browser und Proxies daran zu hindern Dateien doppelt und dreifach zu laden.

Möglichkeiten:
Expires

Das Modul `mod_expires` (expire == ablaufen) setzt zusätzliche HTTP-Header in den Request, der sowohl dem Browser als auch evtl. dazwischen liegenen Proxies sagt, wie häufig sich in etwa diese Datei ändert.

Konkret sieht dies so aus wenn man das Modul bereits eingebunden hat:

Entweder in der Apache-Konfiguration oder einer `.htaccess`-Datei folgende Zeilen einfügen:

```
ExpiresActive On
ExpiresByType image/gif "access plus 1 month"
ExpiresByType image/jpg "access plus 1 month"
ExpiresByType image/png "access plus 1 month"
```

Dieses Beispiel setzt z.B. am 24. Nov. 2007 folgenden Header in den entsprechenden Dateien:

```
Expires: Tue, 24 Dec 2007 20:09:16 GMT
```

D.h. Proxies und der Browser wissen nun, daß es diese Datei einen Monat lang gespeichert halten dürfen, bevor sie es erneut abfragen.

Achtung: es heißt aber nicht, daß sie es auch wirklich tun!

Denn wenn der Speicher überläuft werden meißt die alten Daten rausgeschmissen. Aber es reicht meist für die aktuelle Session des Surfers.

Cache und Proxy

Ein weiteres Problem stellen "non-Public-Sites" dar. Diese erkennen Caches und Proxies vor allem an einem HTTP-Access.

Bei solchen Verzeichnis-Schutz-Systemen werden fast Grundsätzlich alle Dateien neu geladen.

Abhilfe schaft hier das so genannte `public` Attribut des `Cache-Control`-Headers.

Dazu muß aber auch das Modul `mod_headers` aktiviert sein.

```
Header append Cache-Control "public"
```

Header-Tricks im Script

Gerade bei dynamischen Scripten (PHP, Perl, etc.) lassen häufig die Möglichkeiten ausser acht:

Browser schicken manchmal ein `If-Modified-Since`-Header mit. Wenn sich nicht wirklich etwas verändert hat, kann man dies mit einem einfachen `304 Not Modified` ohne Content abhandeln.

Richtig Sinn macht dies aber erst, wenn das Script vorher auch ein `Last-Modified`-Header mit geschickt hat.

Kompression

Meist ist standartmässig das Modul `mod_gzip` aktiviert welches Inhalte Komprimiert sobald der Browser ein `Accept-Encoding: gzip` im Header mit schickt.

Das Problem ist meist, daß dies auf Server-Seite einiges an Performance frißt. Dazu kommen diverse Proxies und Caches, die dann die Headers nicht mehr auslesen können und daher jede Datei mehrfach anfragen.

Daher muß man sich vorher gut überlegen, ob es sinnvoll ist. Z.B. bei statische HTML-Websites.

Alternativ kann man es auch auf Script-Ebene ebenfalls noch einsetzten.

Manuelle Kompression

Bekanntest Beispiel:

Ein großes weltbekanntest Versteigerungs-Portal hat mal seine Performance und Traffic deutlich verändert, indem es alle unnötigen `"` (doppelte Hochkommata) im HTML-Code weggelassen hat.

Was lernen wir daraus?

Reduzierung des Outputs hilft gewaltigt!

Dies fängt bereits bei CSS-Dateien an. Schön Formatiert sind sie gut lesbar, aber alles in einer Zeile ohne Whitespaces nimmt meist deutlich weniger Platz und Traffic ein.

Genauso bei JavaScript und HTML. Und auch PHP/Perl-Scripte: Der Parser muß letztendlich jedes Byte verarbeiten.

Diese Aufgabe kann übrigens auch on-the-fly das Apache2-Modul `mod_blanks` erledigen.

Und noch der altbekannte Tipp: Grafiken so klein wie möglich setzten. Durchaus mal verschiedene Formate ausprobieren. Gerade bei geringerer Farbtiefe sind GIF's deutlich kleiner (und auch schöner) als JPG's. PNG ist auch nicht immer optimal.

Weitere Links:

- Hochleistungs-Apache: Performance-Tuning

- HTML-Tidy
- Skript zum Entfernen der Leerzeichen

Eindeutige ID: #1292
huschi
2008-04-06 20:55