# 5.2 Leistungsfähiges Caching nutzen

Als Caching bezeichnet man die Nutzung eines schnellen Zwischenspeichers (Cache), der dazu dient, häufig angefragte Daten schneller auszuliefern, als dies von der eigentlichen Datenquelle aus möglich wäre. Caching-Mechanismen kommen in verschiedensten Situationen zum Einsatz: Ihre Festplatte hält Daten oft ausgelesener Speicherbereiche in einem kleinen, aber sehr schnellen RAM-Speicher vor. Webbrowser verwalten einen Cache, in dem die Inhalte häufig besuchter Websites auf Ihrem Rechner vorgehalten werden. Somit müssen diese Seiten beim nächsten Aufruf nicht mehr aus dem Netz geladen werden und können schneller dargestellt werden. Auch auf Serverseite lassen sich schließlich mit gezieltem Caching beachtliche Performancesteigerungen erreichen.

Bei den von WordPress ausgelieferten Seiten handelt es sich um dynamische Daten. Diese werden bei jedem Seitenaufruf aus den Template-Dateien des verwendeten Themes mithilfe von Inhalten aus der Datenbank zusammengesetzt.

Rufen zwei Besucher kurz hintereinander die Startseite auf, werden diesen annähernd identische Inhalte ausgeliefert. Dennoch wird für beide Aufrufe die komplette Maschinerie zur Erzeugung dynamischer Inhalte angeleiert. Dies ist ein Punkt, an dem Caching zum Einsatz kommen kann, um den Webserver zu entlasten und die Performance der Website zu steigern.

## Der WordPress Object Cache

WordPress selbst verfügt seit Version 2.0 über einen Mechanismus, um die Ergebnisse aufwendiger Datenbankabfragen in Dateien auf der Festplatte des Webservers zwischenzuspeichern. Der sogenannte *Object Cache* von WordPress lässt sich aktivieren, indem Sie die folgende Zeile zu Ihrer *wp-config.php*-Datei hinzufügen:

- `define(ENABLE_CACHE, true);`

Ist der Object Cache so aktiviert, benutzt WordPress diesen automatisch für seine internen Abfragen. Falls Sie selbst Plug-ins mit Datenbankabfragen entwickeln, können Sie den Cache ebenfalls nutzen, um Ergebnisse aufwendiger Datenbankabfragen oder auch rechenintensiver Funktionen dort zwischenzuspeichern. Das WordPress API stellt dazu die Funktionen *wp_cache_set* und *wp_cache_get* bereit. Ein Beispiel:

```
$data = wp_cache_get('meine_daten', 'container');

if ($data == false) {
   //Falls keine gecacheten Daten gefunden wurden
   //oder der Cache abgelaufen ist, neue Abfrage stellen
   $data = meine_abfrage();
   $expire = 60 * 60 * 24; // Daten für einen Tag cachen
   //Abfrageergebnis in den Cache schreiben
   wp_cache_set('meine_daten', $data, 'container', $expire);
}
```

Der Object Cache identifiziert Daten anhand eines Namens für das Datenpaket und legt diese in einem sogenannten Datencontainer ab. Im Beispiel wurde *meine_daten* als Name für das Datenpaket vergeben, *container* ist der Name des verwendeten Datencontainers. Falls noch kein Datencontainer mit dem angegebenen Namen existiert, legt WordPress diesen automatisch an. In einem Datenpaket sollten immer die Daten der gleichen Abfrage gespeichert werden. Im Beispiel handelt es sich dabei um das Ergebnis der Funktion *meine_abfrage()*.

## Das WP-Cache 2-Plug-in

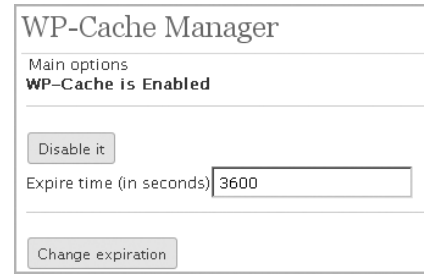
WP-Cache 2 ist ein sehr ausgefeiltes Plug-in, das ganze Seiten einer WordPress-Site in statischen Cache-Dateien hinterlegt, sodass bei einem erneuten Zugriff auf die Seite keine Datenbankzugriffe mehr benötigt werden. Dabei ist das Plug-in intelligent genug, um zu erkennen, wann sich eine Seite geändert hat, zum Beispiel weil ein Besucher einen Kommentar zu einem Artikel abgegeben hat. In diesem Fall gilt die gecachete Version der Seite als abgelaufen (expired), und es wird eine dynamisch erzeugte Seite ausgeliefert.

Das WP-Cache 2-Plug-in kann von der Homepage des Autors Ricardo Galli unter *http://mnm.uib.es/gallir/wp-cache-2/* heruntergeladen werden.

### Konfiguration

Nach der Installation des Plug-ins finden sich unter dem Menüpunkt *Einstellungen/WP-Cache* in der Administrationsoberfläche einige Einstellungsmöglichkeiten für das Plug-in.

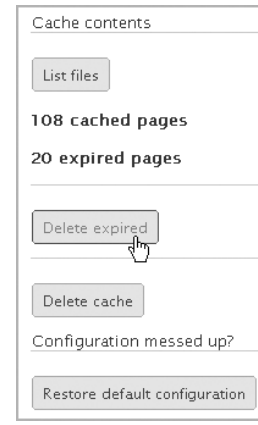
Mit dem Button *Disable it* lässt sich das Plug-in zu Testzwecken vorübergehend deaktivieren.




Die *Expire Time* legt fest, wie lange eine Datei im Cache gültig bleiben soll. Standardeinstellung sind hier 3.600 Sekunden, also eine Stunde. Vorsicht: Wird dieser Wert zu hoch gesetzt, kann das bei einem viel besuchten Blog dazu führen, dass sehr viele Cache-Dateien angelegt werden, die entsprechend viel Plattenplatz belegen. Die Standardeinstellung sollte aber in den seltensten Fällen Probleme bereiten.

Im Feld *Rejected URIs* lassen sich URLs konfigurieren, die vom Caching ausgenommen werden sollen. Standardmäßig ist dieses Feld mit *wp-* vorbelegt, sodass WordPress-interne Dateien wie zum Beispiel die Administrationsseiten nicht gecached werden. Falls man das Datum als Bestandteil der Permalinks konfiguriert hat, kann man mithilfe dieses Feldes auch Artikel vergangener Jahre, die ohnehin eher selten aufgerufen werden, vom Cachen ausschließen, indem man in diesem Feld beispielsweise die Werte */2005/*, */2006/* und */2007/* einträgt.

Im Feld *Accepted files* kann man Dateinamen angeben, die ausdrücklich gecached werden sollen, obwohl sie die Kriterien der *Rejected URIs* erfüllen. Standardmäßig sind hier eine Reihe von Dateien für RSS- und andere Feeds des Blogs eingetragen, die sich bedenkenlos cachen lassen.




Mithilfe des Buttons *List files* kann man sich eine Übersichtsseite aller Seiten im Cache anzeigen lassen. *Delete expired* löscht alle abgelaufenen Dateien im Cache. Diese würden ohnehin beim nächsten Aufruf der betreffenden Seite neu erstellt und belegen effektiv nur Plattenplatz auf dem Webserver. Mit dem Button *Delete cache* lässt sich dagegen der komplette Cache löschen. Dies ist dann hilfreich, wenn man Änderungen am Template oder einem Plug-in vorgenommen hat, die vom WP-Cache-Plug-in nicht erkannt wurden.

Mit dem Button *Restore default configuration* lässt sich das Plug-in wieder auf seine Standardeinstellungen zurücksetzen.

### WP-Cache 2 bei hohem Kommentaraufkommen

Kommt es zeitweise auf einem Blog zu einem extrem hohen Kommentaraufkommen, kann das WP-Cache 2-Plug-in sich kontraproduktiv auswirken. Dies liegt daran, dass bei Eingang eines Kommentars zu einem Artikel die gecachete Version der Seite gelöscht wird, damit der Kommentar beim nächsten Seitenaufruf angezeigt werden kann. Mit einem manuellen Eingriff am Plug-in des Quelltextes lässt sich dieses Problem angehen:

Öffnen Sie dazu die Datei *wp-cache-phase2.php* und kommentieren Sie die folgende Zeile aus:

- `add_action('comment_post', 'wp_cache_get_postid_from_comment', 0);`

Die Kommentare werden so zwar noch in der Datenbank erfasst, aber nicht mehr unmittelbar im Artikel angezeigt. Für die Besucher kann das irritierend sein, es ist aber immer noch besser, als wenn der Server wegen Überlastung zum Stillstand kommt. Nachdem der Cache ausgelaufen ist, werden auch die zunächst durch das Caching verschluckten Kommentare automatisch sichtbar. Natürlich sollten Sie dennoch den auskommentierten Action Hook wieder aktivieren, sobald sich die Situation normalisiert hat.

## WP Super Cache

Das Plug-in WP Super Cache ist ein sogenannter Fork des WP-Cache 2-Plug-ins. Donncha O'Caoimh, der Entwickler dieses Plug-ins, hat den bekannten und erprobten Programmcode von WP-Cache 2 als Grundlage genommen, um ein noch effektiveres Caching-Plug-in zu erstellen.

Der wesentlichste Unterschied zum WP-Cache 2-Plug-in liegt darin, dass WP Super Cache statische HTML-Versionen der gecacheten Seiten anlegt. Diese können dann vom Webserver ausgeliefert werden, ohne dass der PHP Interpreter gestartet werden muss, was bei hoher Last einen entscheidenden Unterschied machen kann. Diese statischen Dateien werden an Besucher ausgeliefert, die nicht angemeldet sind und noch keinen Kommentar hinterlassen haben. Ist ein Besucher als WordPress-Benutzer angemeldet oder hat er

G L O S S A R

**FORK**

Als Fork bezeichnet man in der Softwareentwicklung einen Ableger eines Projekts. Sowohl das Ursprungsprojekt, als auch der Ableger werden unabhängig und meist von verschiedenen Programmierern weiterentwickelt. Nach einiger Zeit können sich dabei das Stammprojekt und der Fork recht drastisch unterscheiden.

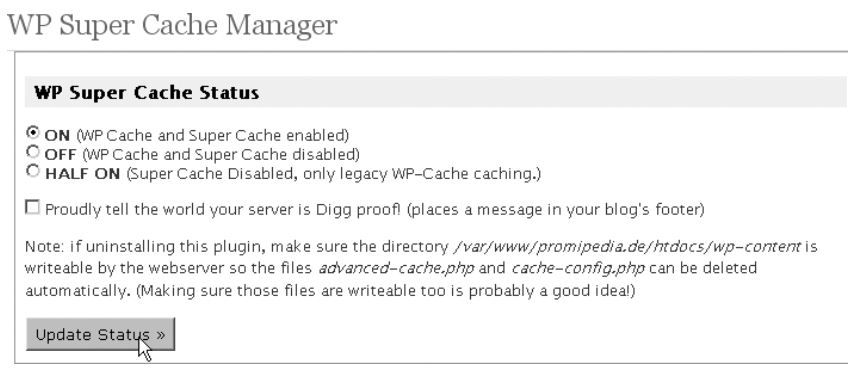
einen Artikel kommentiert, werden ihm dynamische Cache-Seiten, wie sie auch das WP-Cache 2-Plug-in erzeugt, ausgeliefert. Damit wird vermieden, dass benutzerspezifische Daten innerhalb der Seite falsch dargestellt werden.

Sie können das WP Super Cache-Plug-in unter der URL *http://wordpress.org/extend/plugins/wp-super-cache/download/* herunterladen.

Falls Sie das WP-Cache 2-Plug-in installiert haben, sollten Sie dieses vor der Installation von WP Super Cache zunächst deaktivieren und entfernen, da diese beiden Plug-ins sich sonst gegenseitig behindern würden. Deaktivieren Sie dazu zunächst WP-Cache 2 und löschen Sie danach den Ordner *wp-cache* aus Ihrem WordPress-Plug-in-Verzeichnis.

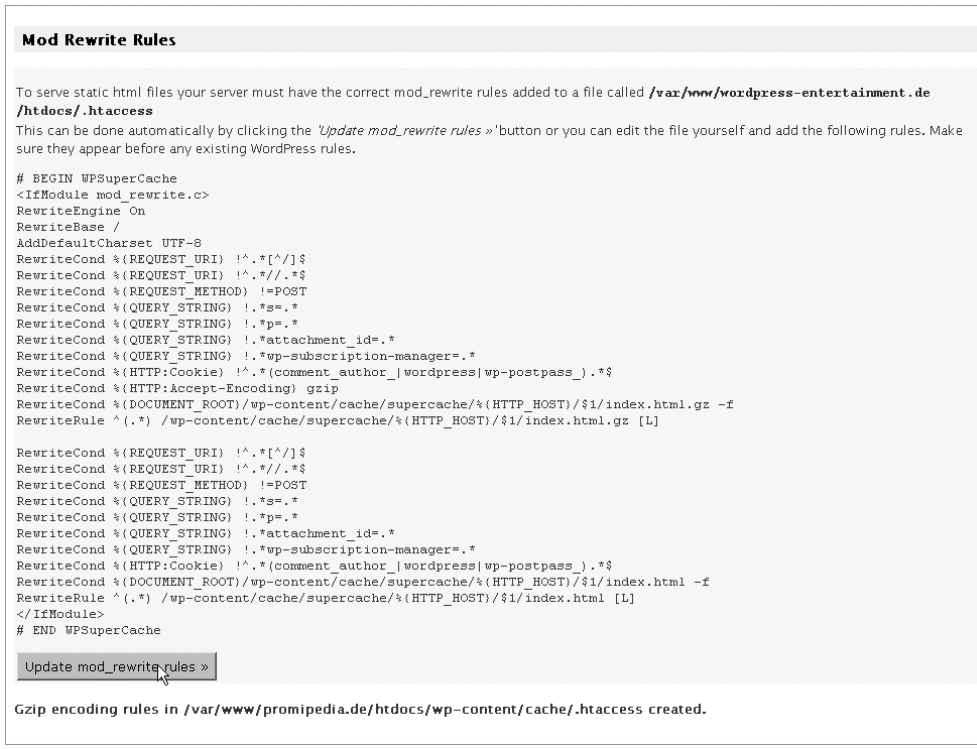
## Konfiguration

Nach der Installation des Plug-ins können Sie WP Super Cache über den Menüpunkt *Einstellungen/WP Super Cache* in der WordPress-Administrationsoberfläche konfigurieren. Bevor das Plug-in seine Arbeit aufnimmt, müssen Sie es dort aktivieren, indem Sie den Status *On* wählen und den Button *Update Status* klicken.




Nach dieser Aktivierung weist das Plug-in darauf hin, dass die Rewrite Rules Ihrer Website aktualisiert werden müssen.

Falls die *.htaccess*-Datei in Ihrem WordPress-Stammverzeichnis für den Webserver schreibbar ist, können Sie diese Änderungen automatisch mit einem Klick auf den Button *Update mod_rewrite Rules* vornehmen lassen. Andernfalls müssen Sie die aufgeführten Regeln aus der Seite kopieren und manuell mit einem Editor in Ihre *.htaccess*-Datei einfügen. Nach Durchführung dieser Änderungen ist das Plug-in bereit, statische Cache-Dateien auszuliefern.



**Mod Rewrite Rules**

To serve static html files your server must have the correct mod_rewrite rules added to a file called **/var/www/wordpress-entertainment.de/htdocs/.htaccess**
This can be done automatically by clicking the *'Update mod_rewrite rules »'* button or you can edit the file yourself and add the following rules. Make sure they appear before any existing WordPress rules.

```
# BEGIN WPSuperCache
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
AddDefaultCharset UTF-8
RewriteCond %{REQUEST_URI} !^.*[^/]$
RewriteCond %{REQUEST_URI} !^.*//.*$
RewriteCond %{REQUEST_METHOD} !=POST
RewriteCond %{QUERY_STRING} !.*s=.*
RewriteCond %{QUERY_STRING} !.*p=.*
RewriteCond %{QUERY_STRING} !.*attachment_id=.*
RewriteCond %{QUERY_STRING} !.*wp-subscription-manager=.*
RewriteCond %{HTTP:Cookie} !^.*(comment_author_|wordpress|wp-postpass_).*$
RewriteCond %{HTTP:Accept-Encoding} gzip
RewriteCond %{DOCUMENT_ROOT}/wp-content/cache/supercache/%{HTTP_HOST}/$1/index.html.gz -f
RewriteRule ^(.*) /wp-content/cache/supercache/%{HTTP_HOST}/$1/index.html.gz [L]

RewriteCond %{REQUEST_URI} !^.*[^/]$
RewriteCond %{REQUEST_URI} !^.*//.*$
RewriteCond %{REQUEST_METHOD} !=POST
RewriteCond %{QUERY_STRING} !.*s=.*
RewriteCond %{QUERY_STRING} !.*p=.*
RewriteCond %{QUERY_STRING} !.*attachment_id=.*
RewriteCond %{QUERY_STRING} !.*wp-subscription-manager=.*
RewriteCond %{HTTP:Cookie} !^.*(comment_author_|wordpress|wp-postpass_).*$
RewriteCond %{DOCUMENT_ROOT}/wp-content/cache/supercache/%{HTTP_HOST}/$1/index.html -f
RewriteRule ^(.*) /wp-content/cache/supercache/%{HTTP_HOST}/$1/index.html [L]
</IfModule>
# END WPSuperCache
```

Update mod_rewrite rules »

Gzip encoding rules in /var/www/promipedia.de/htdocs/wp-content/cache/.htaccess created.

Im Feld *Super Cache Compression* der Konfigurationsseite können Sie die Auslieferung komprimierter Dateien aktivieren, indem Sie diese Option auf *Enabled* setzen. Dies spart Zeit bei der Übertragung vom Webserver zum Browser und reduziert zudem den Traffic-Verbrauch beim Aufruf einer so gecacheten Seite. Leider führt diese Option bei manchen Hostings zu Problemen. In solchen Fällen werden die gecacheten Seiten nach deren Aufruf nicht mehr im Browser angezeigt, sondern stattdessen zum Download auf die Festplatte angeboten. Mitunter lässt sich dieses Problem lösen, indem Sie die folgenden Zeilen zu Ihrer *.htaccess*-Datei hinzufügen:

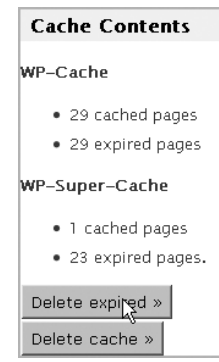
```
<Location /wp-content/cache/supercache/>
    AddEncoding x-gzip .gz
    AddType text/html .gz
</Location>
```

Falls die gecacheten Seiten auch nach dieser Änderung nicht korrekt angezeigt werden, müssen Sie leider auf die Option der komprimierten Auslieferung verzichten. Die Geschwindigkeitsvorteile der statisch vorgehaltenen Webseiten durch WP Super Cache bleiben jedoch dennoch erhalten.

Im Feld *Expiry Time and Garbage Collection* lässt sich wie beim WP-Cache 2-Plug-in festlegen, nach wie viele Sekunden eine gecachete Datei als veraltet anzusehen ist und gelöscht werden soll. Weiterhin kann man auswählen, nach wie vielen Seitenaufrufen das Plug-in die Cache-Dateien nach veralteten Dateien durchsuchen soll. Der Wert reicht dabei von 100 bis 5.000. Wählt man hier einen zu niedrigen Wert, wird unnötig Rechenleistung mit der Suche nach ausgelaufenen Dateien verschwendet. Wird dagegen ein zu hoher Wert eingestellt, sammeln sich jedoch möglicherweise zu viele gecachete Seiten an, was sich ebenfalls negativ auf die Performance auswirkt. Der optimale Wert ist von Site zu Site unterschiedlich und lässt sich nur experimentell ermitteln. In der Praxis hat es sich bewährt, etwa ein Fünftel der täglichen Pageviews der Site für diese Einstellung zu verwenden.

Im Feld *Accepted filenames, rejected URIs* lassen sich analog zum WP-Cache 2-Plug-in bestimmte Dateien vom Cachen ausschließen und Ausnahmeregeln für diese Ausschlüsse definieren.

Im Feld *Rejected User Agents* lassen sich Browserkennungen konfigurieren, mit denen sich Robots von Suchmaschinen zu erkennen geben. Man kann mit dieser Einstellung verhindern, dass diesen gecachete Dateien ausgeliefert werden. Die hier eingestellten Standardwerte können unverändert beibehalten werden.




Auch das Feld *Cache Contents* entspricht seinem Äquivalent im WP-Cache 2-Plug-in. Hier wird unterschieden zwischen dynamisch gecacheten Seiten, deren Anzahl sich unter der Überschrift *WP-Cache* findet, sowie statischen Cache-Dateien unter der Überschrift *WP-Super-Cache*. Die WP-Cache-Seiten sind solche, die für angemeldete Benutzer oder Besucher, die einen Kommentar abgegeben haben, angelegt wurden. Wie bereits vom WP-Cache 2-Plug-in bekannt, lassen sich über die Buttons in diesem Teil der Verwaltung ausgelaufene Dateien (*Delete expired*) oder aber der komplette Cache löschen (*Delete Cache*).

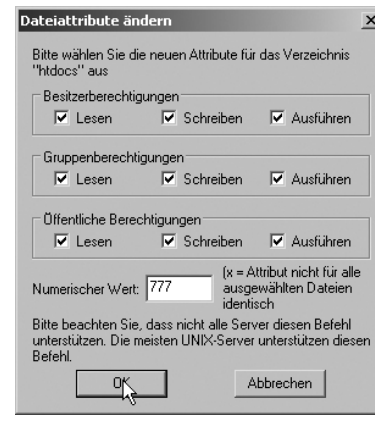
Das nächste Feld der Konfigurationsoberfläche beherbergt den sogenannten *Lockdown Button*. Dieser ist eine wirkungsvolle Rettungsmaßnahme, wenn es zu extremen Lastspitzen kommt, beispielsweise weil das Blog auf der Startseite einer populären Nachrichtenseite steht.

Normalerweise wird die Cache-Datei einer Artikelseite automatisch neu erstellt, wenn ein neuer Kommentar zu dem Artikel eingegangen ist. Wurde der Lock-

down-Modus aktiviert, führen Kommentare nicht zur Neuerstellung der Cache-Datei. Dies kann bei einem plötzlichen, sehr starken Besucheransturm die Rettung für Ihren Server sein. Da allerdings neue Kommentare so nicht mehr sichtbar werden, sollte dieser Modus wieder deaktiviert werden, sobald sich der Ansturm gelegt hat.

Das Feld *Directly Cached Files* erlaubt eine weitere Rettungsmaßnahme, die dann ergriffen werden sollte, wenn ein einzelner Artikel plötzlich einer riesigen Nachfrage ausgesetzt ist. Tragen Sie in diesem Fall den Post Slug des betroffenen Artikels in das Eingabefeld ein und klicken Sie auf den Button *Update direct pages* Button. WP Super Cache legt dann eine statische Version dieses Artikels in Ihrem Dateisystem an, was den Zugriff auf diesen Artikel weiter beschleunigen sollte.


Um diese Funktion zu nutzen, muss der Webserver-Prozess Schreibzugriff auf das WordPress-Verzeichnis haben. Dies können Sie beispielsweise erreichen, indem Sie dem Verzeichnis mit Filezilla die Dateiattribute *777* geben. Vergessen Sie jedoch nicht, die Berechtigung für dieses Verzeichnis nach dem Erstellen der statischen Dateien wieder auf den vorherigen Stand zurückzusetzen, da ein solcher Schreibzugriff ein Sicherheitsrisiko bedeuten kann.




Mit dem Button *Restore default configuration* schließlich erlaubt das WP Super Cache-Plug-in genau wie WP-Cache 2, die Standardkonfiguration des Plug-ins

wiederherzustellen, falls man einmal einen Konfigurationsfehler gemacht hat und diesen nicht mehr selbst korrigieren kann.

### Die Qual der Wahl

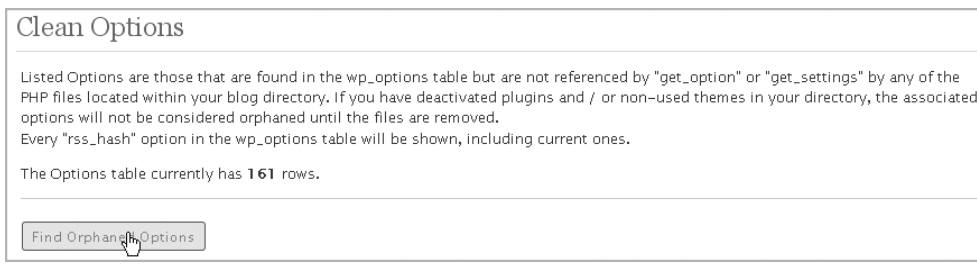
Zwischen WP Super Cache und WP-Cache 2 muss eine Entscheidung gefällt werden. Für Websites, die regelmäßig großen Besucheranstürmen ausgesetzt sind, ist typischerweise WP Super Cache mit seinen Notfallmaßnahmen die sinnvolle Wahl. Auch sonst reduzieren die statischen Cache-Dateien des Super Cache-Plug-ins die Serverlast noch einmal deutlich gegenüber WP-Cache. Andererseits bietet Letzterer für die meisten Websites eine ausreichende Performancesteigerung und ist aufgrund der weniger komplexen Konfiguration oft vorzuziehen.

## 5.3 Plug-ins als Performancebremse

Die WordPress-Kernsoftware arbeitet seit Version 2.0 relativ effizient, was die Nutzung von Systemressourcen und insbesondere Datenbankabfragen angeht. Leider lässt sich dies nicht für alle Plug-ins behaupten, sodass sich ein Blick auf die installierten Plug-ins lohnen kann, wenn die WordPress-Site langsamer als erwartet arbeitet.

### Das Clean Options-Plug-in

Probiert man oft Plug-ins aus, die man danach wieder deinstalliert, bleiben mitunter viele Optionseinstellungen dieser Plug-ins in der Datenbank liegen, da nicht jedes Plug-in mit einer sauberen Deinstallationsroutine ausgestattet wurde. Dies kann auf Dauer dazu führen, dass Datenbankabfragen, mit denen aktive Plug-ins ihre Optionen kontrollieren, ausgebremst werden. Eine Lösung liefert hier das Clean Options-Plug-in, das dazu vorgesehen ist, solche verwaisten Optionseinträge in der Datenbank aufzuspüren und zu eliminieren.



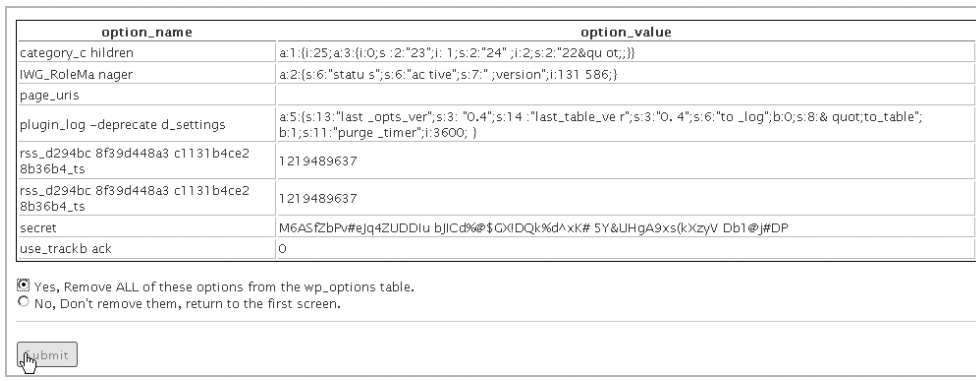
Das Plug-in kann unter *http://wordpress.org/extend/plugins/clean-options/* heruntergeladen werden. Nach der Installation finden Sie den Zugang zu dem Plug-in unter dem Menüpunkt *Werkzeuge/CleanOptions*.

Achtung: Bevor Sie mithilfe dieses Plug-ins Optionen aus Ihrer Datenbank entfernen, führen Sie ein Backup der Datenbank durch. Falls doch durch eine irrtümliche Löschung der falschen Optionen ein Plug-in in seiner Funktion beeinträchtigt wird, können Sie mithilfe der Sicherung leicht wieder einen funktionierenden Zustand Ihres Blogs herstellen.


Nach dem Aufruf dieses Menüpunktes informiert Sie das Plug-in über die Anzahl der Einträge in der Tabelle *wp_options* Ihrer Datenbank. Über den Button *Find Orphaned Options* macht sich das Skript an die Analyse dieser Datenbankeinträge. Sie erhalten dann eine Auflistung von Optionen, die sich keinem derzeit aktiven Plug-in zuordnen lassen. Aus dieser Liste können Sie Optionen auswählen, die Sie dann nach einem Klick auf *View Selected Options Information* am Ende der Seite näher untersuchen können.

Einige dieser Optionen lassen sich möglicherweise nicht auswählen. Dies sind Optionen, die zwar keinem Plug-in zugeordnet werden können, die aber von Clean Options dennoch als aktiv erkannt wurden und nicht gelöscht werden sollten.

Die Folgeseite zeigt Ihnen dann noch einmal die ausgewählten Optionen in einer Übersicht, in der auch zu sehen ist, mit welchen Werten diese belegt sind.



| option_name | option_value |
|---|---|
| category_c hildren | a:1:{i:25;a:3:{i:0;s :2:"23";i: 1;s:2:"24" ;i:2;s:2:"22&qu ot;;}} |
| IWG_RoleMa nager | a:2:{s:6:"statu s";s:6:"ac tive";s:7:" ;version";i:131 586;} |
| page_uris | |
| plugin_log –deprecate d_settings | a:5:{s:13:"last _opts_ver";s:3: "0.4";s:14 :"last_table_ve r";s:3:"0. 4";s:6:"to _log";b:0;s:8:& quot;to_table"; b:1;s:11:"purge _timer";i:3600; } |
| rss_d294bc 8f39d448a3 c1131b4ce2 8b36b4_ts | 1219489637 |
| rss_d294bc 8f39d448a3 c1131b4ce2 8b36b4_ts | 1219489637 |
| secret | M6ASfZbPv#eJq4ZUDDIu bJICd%@$GXIDQk%dAxK# 5Y&UHgA9xs(kXzyV Db1@j#DP |
| use_trackb ack | 0 |

Wenn Sie sich sicher sind, dass alle der ausgewählten Optionen tatsächlich zu bereits entfernten Plug-ins gehören, wählen Sie den Punkt *Yes, Remove ALL of these options from the wp_options table* aus und klicken Sie auf den Button *Submit*. Andernfalls wählen Sie *No, Don't remove them, return to the first screen* und kehren Sie mit dem *Submit*-Button auf die vorhergehende Seite zurück, auf der Sie eine erneute Auswahl treffen können.

## JavaScripts in den Footer verlegen

Viele Plug-ins verwenden JavaScript, um dynamische Elemente zur Seite hinzuzufügen. Ein Beispiel dafür ist das in Kapitel 3 vorgestellte Lightbox-Plug-in, das bei Klick auf ein Bild einen Layer mit der vergrößerten Ansicht des Bildes erzeugt. Oft verbessern solche Skripte die Usability der Website, zur grundlegenden Nutzung der Seite sind sie aber nicht erforderlich. Auch das für den Internet Explorer angepasste hierarchische Menü aus Kapitel 2 ist zur Nutzung der Seite nicht zwingend erforderlich, weshalb der zugehörige JavaScript-Code aus Performancesicht besser im Footer aufgehoben ist.

Oft werden diese Skripte bereits im Kopfbereich der Seite geladen und initialisiert. Der Nachteil hierbei ist, dass der Start des Skripts den weiteren Seitenaufbau ausbremst und die Seite erst dann im Browser dargestellt wird, wenn das Script abgearbeitet wurde. Dadurch entsteht zwar keine zusätzliche Last auf dem Webserver, wohl aber wird die Geduld des Betrachters unnötig strapaziert. Beim Einsatz vieler Plug-ins mit JavaScript-Elementen kann es zu einer merklichen Verzögerung bei der Seitendarstellung kommen.

Wenn Sie den Verdacht haben, dass JavaScript-Plug-ins Ihre Site ausbremsen, lohnt sich zunächst ein Blick in den HTML-Quelltext, um zu sehen, an welcher Stelle JavaScript-Dateien nachgeladen oder JavaScript-Blöcke in den Quelltext geschrieben werden. Passiert dies im Head-Bereich der Seite, ist es einen Versuch wert, diesen Code stattdessen an das Seitenende zu verlagern.

Das Einfügen von JavaScript im Seitenkopf erfolgt typischerweise durch die Registrierung eines Action Hooks für die *wp_head*-Aktion:

```
<?php
   function init_scripts {
   //JavaScript-Dateien nachladen
   echo "<script type=\"text/javascript\"
      src=\"".$plugin_path."js/meinplugin.js\">
```