Drupal 6 Performance Tips

Chapter No.5
"Using DB Maintenance and Boost"
In this package, you will find:
A Biography of the authors of the book
A preview chapter from the book, Chapter NO.5 "Using DB Maintenance and Boost"
A synopsis of the book’s content
Information on where to buy this book

About the Authors

**Trevor James** is a Drupal developer and web designer based in Middletown, MD, USA. Trevor has been designing websites for 13 years using a combination of HTML, XHTML, CSS, and ColdFusion, and has been using Drupal intensively for more than 2 years. Trevor's focus is on building web portals for higher education, public education (K-12), non-profit and small business environments. He is interested in the best methods of developing Drupal themes, Drupal site performance, and using CCK, Views, and Panels to develop frontend interfaces to support data intensive websites. He loves teaching people about Drupal and how to use this excellent open source content management framework.

Trevor has designed and developed websites for many non-profit, education-based, and small business organizations. He is currently working on a number of Drupal-related projects.

Trevor created an 11.5 hour video tutorial series comprising 114 lessons titled *Introduction to Drupal 6* for VTC (Virtual Training Company) in 2009. The video is available via the VTC website here: [http://www.vtc.com/products/Introduction-To-Drupal-6-Tutorials.htm](http://www.vtc.com/products/Introduction-To-Drupal-6-Tutorials.htm)

For More Information:
A huge thank you to my wife, Veronica, and our two beautiful twin girls, Francesca and Clare, for their love and support while I was writing this book.

Thanks to my father-in-law, Tony Gornik, for offering his residence in Hershey, PA, as writing space on weekends. The quiet and slower pace of Hershey helped inspire the writing of these chapters. Many thanks to the Hershey Fire Company crew for giving me and my twin daughters tours of the big trucks during the much-needed writing breaks.

Many thanks to the entire Packt editorial and project team for inviting me to work on this project and for continuing to publish excellent titles on Drupal and open source applications. More thanks to Steven Wilding, Packt Acquisition Editor; Joel Goveya, Project Coordinator; Akash Johari, Technical Editor; Lata Basantani, Projects Team Leader; and Patricia Weir, for keeping the project on track and for guiding me in the construction of this title. I look forward to working with you all again in the near future.

**TJ Holowaychuk**, the president of [http://vision-media.ca](http://vision-media.ca), is a self-taught web development guru whose skills range from high performance programming in C to agile and elegant solutions written in Ruby or PHP. He has contributed to and started over 50 open source projects including Drupal, JSpec, Evolution CMS, and jQuery. With such a large array of skills, TJ provides a unique perspective with all challenges regarding performance, design, or development.

For More Information:
Drupal 6 Performance Tips

More The Drupal content management framework allows us to get a website up and running quickly, and proves that a multi-layer website and application environment doesn't need to be complex to set up and configure. The next step after we get our site installed, themed, and populated with content, is to monitor our site's performance. We, as users of the site and developers of the site's architecture and backend, want our site to run smoothly and quickly. We want our page loads to be super quick and our backend administration to run lightning fast. How do we optimize our large Drupal-powered, database-driven, content-heavy website with performance and speed in mind? This book will show you the steps to enable the performance 'boost' on your Drupal site.

We will discuss all aspects of Drupal performance from simple optimization and site maintenance to the larger and more complex issues of anonymous and authenticated site caching. We'll look at some basic core Drupal modules that help to govern and control performance on our site, and also look in detail at more advanced contributed module options, such as the Development, Boost, Authcache, Advanced Cache, and Cache Router modules.

With speed in mind, both for our anonymous site visitors and our logged in users, we're going to take a close look at how to optimize our Drupal site for higher performance. This book is an introduction to this complex and large subject, and we hope that it serves as a stepping stone for both novice and advanced Drupal users and developers.

What This Book Covers

Chapter 1, Upgrading Drupal, focuses on preparing a Drupal environment for running a high performance Drupal website. We will discuss upgrading Drupal 5.x to Drupal 6.x, creating backups of our site files and databases, running the Drupal Status report, upgrading contributed modules to their latest 6.x versions, and running update.php. We will also tweak our PHP settings using the Drupal settings.php file.

Chapter 2, Maintaining your Drupal Site, covers the basics of maintaining your Drupal website including inspecting your Drupal configuration file, checking your MySQL and PHP configurations, enabling and using the Drupal Update Status module, disabling and uninstalling contributed modules, and clearing the Drupal performance cache and theme registry. We'll also look at running cron jobs, tweaking our php.ini file, and tweaking our .htaccess file.

For More Information:

Chapter 3, Using Development Modules and Tools, focuses on using Drupal development modules and tools such as the Development module. We will look in detail at the Development module's functionality and use it to monitor performance on our site.

Chapter 4, Performance Optimization, focuses on Drupal performance optimization, including throttling modules and blocks through the Development module to generate dummy taxonomy, and content for our site using Views 2.x, and clearing our Views cache.

Chapter 5, Using DB Maintenance and Boost, focuses on using the DB Maintenance and the Boost modules. We'll look in detail at a basic configuration of the Boost module to enhance performance for our anonymous site users.

Chapter 6, Advanced Boost, focuses on using the Boost module to do advanced performance functionality. We'll look at using the Boost module along with Global Redirect and Transliteration, configuring advanced Boost module caching, configuring Boost crawler, and how to check and tweak our Boost .htaccess settings.

Chapter 7, Using Memcache API and Integration, focuses on using the Memcache API and Integration module. We will install a development WAMP environment using MoWeS Portable in this chapter as well as install the Memcached binary libraries, integrate and configure Memcached to work with PHP, and test the module on our development site.

Chapter 8, Advanced Caching and Contributed Modules for Caching, focuses on Advanced Drupal caching and using contributed modules for caching on our site. We will discuss using the Cache Router, Authcache, and Advanced Cache modules. Chapter 9, Multisite Configuration and Performance, focuses on Drupal multisite configuration and performance. We will create multisite folders and configure our Drupal settings.php for multisite. We will tweak our httpd.conf file to support multisite, and use caching in a multisite environment.

For More Information:
In this chapter, we're going to install, configure, and utilize the DB Maintenance and Boost modules. DB Maintenance will allow us to maintain and optimize our MySQL database from within the Drupal admin interface. Boost will help us to speed up page load times on our site for our anonymous site users by using its sophisticated and advanced page, HTML, CSS, and JavaScript caching mechanisms. We'll look at basic introductory Boost concepts in this chapter, and later in Chapter 6 we will look at more advanced topics using the Boost module. Consider this a two part dose of the Boost module. Both of these contributed modules will help you diagnose problems on your site and server as well as help to keep your site running smoothly and in an optimized fashion. These are not required modules, but rather are recommended modules to add to your Drupal performance arsenal. The way this chapter will work is that we'll outline the purpose of each module, install and configure it, and then use it on a specific topic, for example, within your site. This will give you some practice using contributed Drupal modules and also a look at the variety of performance-based modules that are available from the Drupal project community.

By the end of this chapter you will know how to install, configure, and use the following contributed performance modules:

- DB Maintenance module
- Boost

For More Information:

Using DB Maintenance and Boost

Using the DB Maintenance module

The DB Maintenance module can be used to optimize your MySQL database tables. Depending on the type of database you are running, the module allows you to use a function called OPTIMIZE TABLE, which troubleshoots and then optimizes various errors in your MySQL tables. For MyISAM tables, the OPTIMIZE TABLE will repair your database tables if they have deleted rows. For BDB and InnoDB types of tables the function will rebuild the entire table. You can use this module in tandem with phpMyAdmin to determine if you do or do not need to optimize your database tables. The benefit of this module is that it allows you to keep your database optimized and defragmented, similar to keeping your computer hard drive optimized and defragmented so that it runs faster, and you can do all this from the Drupal administrative interface.

The project page where you can download the module is here: http://drupal.org/project/db_maintenance.

Download the module tar.gz and extract it to your desktop. Then, upload the files through FTP, or upload and extract using a cPanel utility if your host provides this. The module should go in your /sites/all/modules directory.

Once you upload and extract the module folder, enable the module on your modules admin page and save your configuration. We’ll use the version that’s recommended for Drupal 6.x, which is 6.x-1.1. You can try out the beta version, but you should not run this beta version on a production level website unless you’ve tested it sufficiently in a sandbox environment.

Once you save your module configuration, you’ll notice that the module adds a link to its settings and configuration page under your main Site configuration section. Go to Site configuration | DB maintenance to access the configuration admin screen for the module. The DB maintenance screen will contain a checkbox at the top allowing you to log OPTIMIZE queries. If you check this box, your watchdog log entries module will log all table optimization entries and give you detailed information on the tables that were optimized.
At the time of writing this book, the 1.1 version of the DB Maintenance module contained bugs that caused glitches with the method of adding this module’s queries to the recent log entries or prevented this entirely. You may also experience these glitches. The module’s developers are aware of the issues because they have been posted to the issue queue at http://drupal.org/ on the module project page.

Let's go ahead and check this box. You can then select the frequency with which you would like to run the optimization. The choices are daily, Run during every cron, Hourly, Bi-Hourly, Daily, Bi-Daily, Weekly, Bi-Weekly, Monthly, and Bi-Monthly. You can also click on the Optimize now link to force the optimization to occur immediately without scheduling in advance. We'll click on this link for the purpose of this demo, but in future you may want to schedule the optimization. We'll then run a cron job through the Status report, or a module such as Poormanscron, and the tables will be optimized.

Next, you can select the tables in your Drupal database that you want to optimize. A nice feature of this module is that it allows you to multi select database tables, only select a few tables, or just one table. This gives you the same flexibility and functionality as your phpMyAdmin tool, but you can run everything from within your Drupal interface. It's like a phpMyAdmin lite version right in your Drupal site. This is a preferred option for those developers who may not have immediate access to a client's phpMyAdmin or a host's database management utility.
Using DB Maintenance and Boost

Choose a selection of tables that you want to optimize, or select all the tables. For this demo I'm going to optimize all of my content type tables, so I'll select all of those. I'll also optimize my block tables:

blocks
blocks_roles
category
content_type_blog
content_type_book
content_type_forum
content_type_page
content_type_photo
content_type_poll
content_type_story
content_type_webform

Once you've selected the tables you want to optimize, click on the Optimize now link.

As with any module or optimization enhancement that you make to your Drupal site, it is good practice to run a full backup of your MySQL database before performing any maintenance, including optimizing tables using the DB Maintenance module. This way you will have a full backup of your data if you run into any issues that the module could potentially create. It's better to play it safe and perform the backup first.

Once you click on the Optimize now link, you should receive a message notifying you that the Database tables are optimized.

This concludes our discussion and walkthrough of using the DB Maintenance module. Let's now turn to the Boost module and use it to speed up our site page and content loads.
Using the Boost module

We're going to turn our attention to the Boost module in this section. Boost is a contributed module that allows you to run incredibly advanced static page caching on your Drupal site. This caching mechanism will help to increase performance and scalability on your site, especially if it gets heavy traffic and anonymous page visits, and it is on a shared hosting environment. This is usually the first contributed performance-based module to turn to for help when you host your Drupal site on a shared server. Developers running Drupal sites on shared servers and running sites that serve predominantly anonymous Drupal users will definitely want to try out this module. It's also a fun module to use from a technical standpoint because you can see the results immediately, as you configure it.

The Drupal project page for the module is here: http://drupal.org/project/boost. There is a wealth of detailed information about the module on this project page, including announcements about upcoming conference presentations that focus on the Boost module, testimonials, install instructions, and links to documentation and associated modules that you may want to run alongside Boost. It is very popular and has quite a following in the Drupal development community. I definitely recommend reading about this module and all of its install and configuration instructions in detail before attempting to use it.

The install paragraph suggests reading through the module README.txt file before running the install for details on how the module works. There are also detailed instructions and documentation on the module here: http://drupal.org/node/545664.

Note that the one requirement to use this module is that your Drupal site must have clean URLs configured and enabled. It's a good idea to make sure you are running clean URLs on your site before you start installing and configuring Boost.

Additionally, there are some recommended modules that the developers encourage you to install in tandem with the Boost module. We will install two of these modules: Global Redirect and Transliteration. The Global Redirect module runs a number of checks on your website including the following:

- Checks the current URL for a Drupal path alias and does a 301 redirect to the URL if it is not being used.
- Checks the current URL for a trailing / and removes the slash if it's present in Drupal URLs.
- Checks if the current URL is the same as the site's front page and redirects to the front page if it locates a match.

For More Information:
Using DB Maintenance and Boost

- Checks to see if you are using clean URLs. If you do have clean URLs enabled, this module ensures URLs are accessed using the clean URL method rather than an unclean method (for example, ?q=user).
- Checks access to the URL. If a user does not have permissions to view the URL, then no redirects are allowed. This helps to protect private URL aliases.
- Checks to ensure the alias matches the URL it is aliasing. So, if you have a URL alias such as /about and this directs to node/23, then a user on your site can access the page using either of those URLs.
- The Transliteration module removes white space and non-ASCII characters in your URLs. For example, it will try and add underscores to fill white space in a URL.

Installing and enabling these two modules will help remove glitches and errors in your site's path structure.

If you haven't already, we'll also take the time now to install the Poormanscron module and set up and configure automatic cron runs instead of having to continue running cron manually. We'll return to installing and configuring Poormanscron later in this chapter, but just keep it on your radar for now.

Let's go ahead and install the Boost module and take a closer look at some of its features.

**Installing and configuring Boost**

Installation and configuration instructions are provided on drupal.org and also in the module's README file. Read these in detail before installing and enabling the module: http://drupal.org/node/545908.

Additionally, the module notes specify that the module will install and enable using smart defaults and these should work fine in most shared server environments. We'll look at the default settings in this chapter and how to make more advanced configurations in Chapter 6.

For More Information:
Follow these steps to install the module(s):

- Download the **Boost** module along with **Transliteration**, **Global Redirect**, and **Poormanscron** to your desktop. Unzip them and then upload to your `/sites/all/modules` directory. The latest version of Boost is **6.x-1.13**, **Transliteration** is **6.x-2.1**, and **Global Redirect** is **6.x-1.2**.

- Check to make sure you have clean URLs enabled and working correctly in your site at the settings page here: `/admin/settings/clean-urls`.

- Go to your modules admin list and enable the **Boost**, **Transliteration**, **Global Redirect**, and **Poormanscron** modules. Boost will be in its own Caching section of the modules list.

Once enabled, you will see a series of messages loading on the modules page telling you that:

- **Existing filenames have not been transliterated.**
- **Boost has been successfully installed.** There will be a link to the module configuration settings page and a notification telling you that two blocks can be enabled to help you administer Boost, as well as a block to support stats.

It's nice that the Transliteration module informs you that it has not changed the URL paths of your previously posted Drupal pages. It's only going to affect new pages, and only if you enable and configure the module.

Click on the configuration settings link to launch the Boost admin page. You'll notice that the Boost configuration page is a tab that's part of the overall Drupal Performance admin section of the site. You can also get to the configuration page by going to **Site configuration | Performance | Boost Settings**.

**Boost settings**

Boost settings are split up into the following sections:

- **Boost File Cache**
- **Boost cacheability settings**
- **Boost directories and file extensions**
- **Boost advanced settings**
- **Boost crawler**
- **Boost Apache**
- **Clear Boost's Database**

For More Information:

All of these sections and settings are explained in detail on the Drupal module project Installation and Settings page here: http://drupal.org/node/545908. We're going to look at most of these configuration settings in detail.

**Boost File Cache settings**

Most of the default settings will work for us here, but let's run through them all. We want to make sure that we have **Boost – Static page cache** set to **Enabled**. What this will do is store all of our Drupal-generated nodes and pages as static HTML files in a special cache directory in our Drupal site directory. Caching pages will help our site to deliver its pages and content in the fastest possible manner without turning to PHP or Drupal. This will provide us with improved performance, but you need to bear in mind that this type of caching will mostly benefit anonymous users of your website and not logged-in authenticated users who will be depending more on your Drupal functionality. In some cases, if you have a site that functions on multiple levels (for anonymous and authenticated users), it will be a trade-off to use this module, it is more advantageous to use it on sites that have more anonymous user access. On popular sites that allow mostly anonymous user visits, this module is a necessity.

**Gzip page compression** setting should be set to **Disabled** for our example. Page compression is normally handled by the Apache web server itself and you will not need to enable any additional page compression. If Apache is compressing pages, this can actually interfere with the Drupal page compression and crash your website, so do not enable this until you are sure Apache or the web server you are using is not performing page compression.

We'll leave the **Boost – HTML – Default minimum cache lifetime** set to **1 hour** for **HTML**, **XML**, and **JSON**.

You will also notice that you can easily clear your Boost cached data by clicking on the **Clear ALL Boost cached data** button, and your expired cached data by clicking on that button. Your screen should look something like this:

---

**For More Information:**

Boost cacheability settings

Scroll to this section and leave the defaults enabled—this includes **Cache pages that contain URL Variables** and **Cache HTML documents** that will cause Boost to cache all content in your Drupal pages. You can choose here to **Cache your XML and JSON**. When you select cache XML and JSON, the corresponding **Boost - XML** and **Boost - JSON** minimum cache lifetimes will be selectable in the **Boost File Cache** section. Let's go ahead and check the box to cache our XML and RSS feeds.

Let's leave the **Cache .css** and **Cache .js** checked, so our CSS and JavaScript files are also cached.
Using DB Maintenance and Boost

Optionally, you can specify which pages on your site to cache by adding the page URLs to the Pages box and then by specifying that you only want to cache those pages. You can also enable PHP code here if you have specific PHP code snippets you want to run. This is similar to how the core Drupal Block configuration works.

You can see that the module developers spent a lot of time working out the best workflow for the configuration so that it matches other core Drupal module configuration pages in usability and workflow.

At this point your screen should look similar to this:

<table>
<thead>
<tr>
<th>Boost cacheability settings (key=cacheability, weight=0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Cache pages that contain URL Variables (key=boost_cache_query, weight=0)</td>
</tr>
<tr>
<td>- Cache html documents/pages (key=boost_cache_html, weight=0)</td>
</tr>
<tr>
<td>- Cache xml &amp; feed (key=boost_cache_xml, weight=0)</td>
</tr>
<tr>
<td>- Cache json (key=boost_cache_json, weight=0)</td>
</tr>
<tr>
<td>- Cache css (key=boost_cache_css, weight=0)</td>
</tr>
<tr>
<td>- Cache js (key=boost_cache_js, weight=0)</td>
</tr>
</tbody>
</table>

**Statically cache specific pages (key=boost_cacheability_option, weight=0):**

- Cache every page except the listed pages.
- Cache only the listed pages.
- Cache pages for which the following PHP code returns TRUE (PHP-mode, experts only).

**Pages (key=boost_cacheability_pages, weight=0):**

Enter one page per line as Drupal paths. The "*" character is a wildcard. Example paths are "blog" for the blog page and "ajax/*" for every personal blog which is the front page if the PHP-mode is chosen, enter PHP code between \\php \\ tags. Note that executing incorrect PHP-code can severely break your Drupal site.

### Boost directories and file extensions

This is a very important section. Here you tell Boost module where to store your cached data and files. The `README` file also explains this section in detail and the corresponding tweaks we will eventually run to our `.htaccess` file to enable Boost to work.

---

**For More Information:**

We need to create a folder in our site's directory for our cached material. Our folder will be called `cache`, but it needs to exist on the site. So if this folder and the subsequent path folders do not exist, you need to create them and make sure they are writeable on the web server. The `/cache` folder should be created in the root directory of your Drupal site. Create a folder called `cache` at the root of your Drupal installation alongside your other Drupal folders such as `/files`, `/includes`, `/misc`, and so on. Make sure the `/cache` folder is writeable. This means setting your permissions for the `/cache` folder to 777. Drupal or the Boost module may create the folder automatically if you do not create the folder in advance, but it's a good idea to create it anyway.

You can create the cache folder through FTP or a file manager utility like cPanel. The directory we create must be named `cache`, and the Cache file path is set to `cache/normal/variantcube.com/fire`. Then make sure that this directory and its sub folders are created on the site and are writeable. So I'll go ahead and create the `cache`, `normal`, `variantcube.com`, and `fire` folders.

We'll return to this discussion during our multisite chapter because we can tell the Boost module not to store cache file paths in the database. This is helpful in multisite installations when you are running Boost.

Your screen should look something like this:

```plaintext
- Boost directories and file extensions (key=directories, weight=0)

  Cache Dir (key=boost_root_cache_dir, weight=0): 
  cache

- Do not store the cache file path in the database (key=boost_multisite_single_db, weight=0)
  Enabling will allow for correct multi-site caching, in cases where different content is served from the same Drupal installation, based on domain. Examples: Multi-site with a shared/anonymous database, site translation detection based on domain, and the contributed "Domain Access" module.

  Cache file path (key=boost_file_path, weight=0): 
  cache/normal/variantcube.com/fire

  A file system path where the static cache files will be stored. This directory has to exist and be writable by Drupal. The default setting is to store the files in a directory named cache/normal/variantcube.com/fire under the Drupal installation directory. If you change this, you must also change the URL rewrite rules in your web server configuration (htaccess for Apache, lighttpd.conf for Lighttpd), or caching will not work.

- Generated output storage (HTML, XML, AJAX) (key=generated, weight=0)

- Static storage (CSS, JS) (key=static, weight=0)
```

For More Information:
You can also specify storage locations for your HTML, XML, and AJAX output, and static storage for your CSS and JS files if you want to keep those in a specific location of the cache folder. I will leave these as the default settings for our examples and let Boost decide where it wants to store each set of files within the path and directory structure I’ve created.

Before moving on to enabling the Boost default admin blocks, save the configuration and then create the .htaccess file. First, save configuration on your main Boost settings configuration page. This will save the configuration work we've just enabled. Having done this, let's move on to make a required tweak to our .htaccess file.

**HTACCESS file tweaks**

There is one important tweak that we need to carry out on our .htaccess file. We need to do this in order to make our Boost configuration work. So far in this section on the Boost module, this is the most complex configuration step we've taken. Follow these steps in order to tweak your .htaccess (this is also explained in detail in the README file in the module folder):

1. Back up your original .htaccess file in your Drupal install directory so that you have a backup in case of problems.
2. Copy the custom generated htaccess rule from **Administer | Site configuration | Performance | htaccess rules generation** or by clicking on the tab in your Performance section that says **Boost htaccess rules generation**.
3. When you click on the button, you’ll see a text box with the rules presented as a big block of code. Copy this code and paste it into your .htaccess file (through editing mode in either FTP or cPanel). The module help text here tells you to copy this rule and paste it below the # RewriteBase / and above the # Rewrite URLs of the form 'x' to the form 'index.php?q=x'.

So you should have something in your current .htaccess that looks like the following:

```bash
# RewriteBase /
----------------- paste the rules here -----------------
#Rewrite URLs of the form 'x' to the form 'index.php?q=x'.
```

4. Paste the code in the position indicated paste the rules here. Make sure you look for the # RewriteBase / commented line of code and then paste in your rewrite rules immediately following that comment line. Go ahead and do this now.

For More Information:

The resulting code looks like this:

```apache
### BOOST START ###
AddDefaultCharset utf-8
<FilesMatch "\.(html|\.xml)$">
  <IfModule mod_headers.c>
    Header set Expires "Sun, 19 Nov 1978 05:00:00 GMT"
    Header set Cache-Control "no-store, no-cache, must-revalidate,
    post-check=0, pre-check=0"
  </IfModule>
</FilesMatch>
<IfModule mod_mime.c>
  AddCharset utf-8 .html
  AddCharset utf-8 .xml
  AddCharset utf-8 .css
  AddCharset utf-8 .js
</IfModule>
<FilesMatch "\.(html|\.xml)$">
  ForceType text/html
</FilesMatch>
<FilesMatch "\.(xml|\.css|\.js)$">
  ForceType text/css
</FilesMatch>
<FilesMatch "\.(js|\.css|\.html)$">
  ForceType text/javascript
</FilesMatch>
# Gzip Cookie Test

# NORMAL - Cached css & js files
  RewriteCond %{DOCUMENT_ROOT}/cache/perm/%{SERVER_NAME}%{REQUEST_URI}_\.css -s
  RewriteRule .* cache/perm/%{SERVER_NAME}%{REQUEST_URI}_\.css [L,QSA,T=text/css]
  RewriteCond %{DOCUMENT_ROOT}/cache/perm/%{SERVER_NAME}%{REQUEST_URI}_\.js -s
  RewriteRule .* cache/perm/%{SERVER_NAME}%{REQUEST_URI}_\.js [L,QSA,T=text/javascript]

# Caching for anonymous users

For More Information:
```
# Skip boost IF not get request OR uri has wrong dir OR cookie is set OR request came from this server OR https request
RewriteCond %{REQUEST_METHOD} !^GET$ [OR]
RewriteCond %{REQUEST_URI} (^/fire\(admin|cache|misc|modules|sites|system|themes\|node\|add\))|(/(comment\|reply\|edit\|user\|user/ (login\|password\|register)\))$) [OR]
RewriteCond %{HTTP_COOKIE} DRUPAL_UID [OR]
RewriteCond %{REMOTE_ADDR} ^74\.|220\.|207\.|144$ [OR]
RewriteCond %{HTTPS} on
RewriteRule .* - [S=2]

# NORMAL
RewriteCond %{DOCUMENT_ROOT}/fire/cache/normal/%{SERVER_NAME}%{REQUEST_URI}_%{QUERY_STRING}.html -s
RewriteRule .* cache/normal/%{SERVER_NAME}%{REQUEST_URI}_%{QUERY_STRING}.html [L,T=text/html]
RewriteCond %{DOCUMENT_ROOT}/fire/cache/normal/%{SERVER_NAME}%{REQUEST_URI}_%{QUERY_STRING}.xml -s
RewriteRule .* cache/normal/%{SERVER_NAME}%{REQUEST_URI}_%{QUERY_STRING}.xml [L,T=text/xml]

### BOOST END ###

Once it's pasted in my .htaccess file, I should see the code start below my
#RewriteBase / and it will look like this (this is just an excerpt):

# RewriteBase /
### BOOST START ###
AddDefaultCharset utf-8
<FilesMatch "\.(html|xml)$">
  <IfModule mod_headers.c>
    Header set Expires "Sun, 19 Nov 1978 05:00:00 GMT"
    Header set Cache-Control "no-store, no-cache, must-revalidate, post-check=0, pre-check=0"
  </IfModule>
</FilesMatch>
<IfModule mod_mime.c>
  AddCharset utf-8 .html
  AddCharset utf-8 .xml
  AddCharset utf-8 .css
  AddCharset utf-8 .js
</IfModule>
<FilesMatch "\.(html|gz)$">
  ForceType text/html
</FilesMatch>
<FilesMatch "\.(xml|gz)$">

For More Information:
Refresh your Performance settings page in Drupal. That's it! You're now ready to use the Boost module and start speeding up your page loads. Let's start using it as an anonymous user on our site.

Testing your Boost configuration

Now we're going to test out our Boost configuration and make sure everything is working with our initial basic settings and the .htaccess configuration that we're running. Log out of your website in your current browser or open up another web browser so that you can browse around your site as an anonymous user.

The main thing we want to check on is that our static HTML type files (our Drupal pages or nodes) are being cached and stored in the cache directory we have specified in the module configuration. If we chose to use a GZIP compression, we will want to check to make sure the ZIP files are being generated and stored.

Also, run your Status report and view your log entries to check to see if any errors related to the module configuration are being thrown.

You should start noticing a performance boost on your site immediately, as you browse around your site. Start clicking around and opening different nodes on your site and admire the faster performance! You should notice it.

If we check the cache directory on our site, we should notice that the Boost module has started writing HTML files to our cache directory. In the directory you should now see the following folders:

/cache/normal/variantcube.com/fire/node

For More Information:

Boost has automatically created a new folder called `/node` where it will store the cached HTML versions of the Drupal pages it loads. For example, if we look into our `/node` directory, we should see a bunch of HTML files that have been cached while we've browsed anonymously on our site. You can almost see this happen in real time if you browse to a page and then immediately refresh your remote server/site window in your FTP client (while in the `/node` folder). I see the following files corresponding to their Drupal nodes:

- `201_.html`
- `202_.html`
- `203_.html`
- `206_.html`
- `208_.html`

These correspond to:

- `node/201`
- `node/202`
- `node/203`
- `node/206`
- `node/208`

Also, at the root of our `/fire` directory, we should see any non-node pages (for example, pages created using Drupal Views module). In our case, our main Photo gallery View page has been cached: `photo_gallery.html`. This page corresponds to our `photo_gallery` View page.

You can immediately see the power and flexibility of this module by inspecting your cache directory.

You should notice a performance increase on all of these cached pages because the pages that are loading are now your Boost-powered HTML pages. So, multiple clicking on one Drupal node should demonstrate how quickly your pages are now loading.

The module has created another folder in your `/fire/cache` directory called `perm`. The `/perm` folder contains your CSS and JS files as they are cached. If you look in this folder, you'll see paths to the following folders:

- `/cache/perm/variantcube.com/fire/files/css`
- `/cache/perm/variantcube.com/fire/files/js`

If you look in your CSS directory, you should see cached versions of your CSS files, and if you look in your `/js` directory, you should see a cached version of your JavaScript.

For More Information:

Another method of checking the module is working correctly is to view source on your pages (by viewing source in your web browser) and see if the following code is being added to your HTML output:

```html
<!-- Page cached by Boost @ 2009-10-23 13:56:03, expires @ 2009-10-23 14:56:03 -->
```

So the actual HTML source in the web browser will tell you that you are viewing a cached version of the page rather than a dynamically generated version of the page. It also tells you when this cached page version will expire—based on our configuration, basically one hour after it's been loaded depending on our Boost module settings.

Everything appears to be working fine with our initial Boost installation and configuration. Sit back and behold the power of Boost!

**Boost and Poormanscron**

Checking our Status report will show us that we're running an incorrect version of Poormanscron. Boost is optimized to work with the latest dev or 2.0 branch of Poormanscron. So let's go ahead and install the latest version so that our cron runs will work correctly with Boost.

Visit the Poormanscron project page and download the 6.x.-2.0-beta1 release and extract and upload it to our /sites/all/modules directory. Then run your Status report again to check to make sure the Boost warning has disappeared. You may need to run your update.php script, as this module update will make changes to your database schema. Run update.php and then refresh your Status report.

In your Status report, you should now see the Boost row state: **Boost Installed correctly, should be working if properly configured.**

For More Information:

Using DB Maintenance and Boost

Configuring Poormanscron

The updated 2.x-beta1 version of Poormanscron is the precursor module to the eventual Drupal 7 core cron functionality. In Drupal 7, the functionality of the Poormanscron module will be part of the default core processes. For this reason the beta1 version does not give you a module configuration page. It will just run cron automatically, based on a setting on your Site information page. Go here to see that setting: Site configuration | Site information. Now you have an automatically run cron setting that you can select from. We'll use the default 1 hour cron run. This is a nice preview of some of the new built-in functionality of Drupal 7 core.

Clearing the Boost cache

If you want to clear your Boost cache at any time and see how this affects the /cache folder on your site, go to the Boost cache configuration page at Performance | Boost Settings and click on the Clear ALL Boost cached data button and the Clear Boost expired data button. Since you started caching, these buttons will tell you how many pages they are caching respectively. This is another good example of the flexibility of this module.

Once you have cleared cached data from Boost, you will receive messages telling you that the Boost: Static page cache cleared and/or a message stating Boost: Expired stale files from static page cache. To see the results, check your /cache directories and you should see that all the previously cached HTML pages are now deleted. Through FTP you may need to run your F5 function key to refresh the /node directory. The HTML files will be deleted. If you refresh your /node directory, it should now be empty. This is a method of clearing the cache manually instead of waiting for it to clear based on the minimum cache lifetime you set as per cron runs. Again, this is a very flexible functionality allowing you both manual control and auto control of your page caching.

Boost admin and stats blocks

The Boost module provides you with three blocks that you can enable and use on the administrative side of the site. Go to Site building | Blocks and look for the following blocks:

- Boost: AJAX core statistics
- Boost: Pages cache configuration
- Boost: Pages cache status

For More Information:

Enable all three of these blocks to show in the right sidebar of your site. You can configure each block and choose to show it only on specific pages of your site and for specific roles as well, such as site admin, if you prefer. Configure each block and then enable in your right sidebar. I’m also going to make sure my two new blocks are showing at the top of the right sidebar area above any other blocks I may have configured. I’ll put the status block first and the cache configuration block second in weight order.

Now, in your right sidebar on each page, when you are logged in as an admin user you should see both the Pages cache configuration block and the Pages cache status block. In another browser (if you haven't already), browse around your website as an anonymous user. Now load those same pages as an authenticated admin user and view the blocks and the information they are showing you about each page.

**Boost: Pages cache status block**

This is what the *Boost: Pages cache status block* should look like (or similar to) in your site. Here I’m on node/203 of our site and the block looks like this:

The status block tells us whether the site is being served to anonymous users as a static cached HTML page or whether it’s still being served 'live' as a dynamic Drupal page. This is very helpful if you have thousands of pages on your site and you're trying to determine if the page is being served static or live. You can then determine quickly if you need to enable caching on that specific page (if you have a more complex Boost configuration where you're only using Boost to serve specific pages).
This block also tells us how many seconds it takes to generate the cached page. Here, on node/203, it takes 2.186 seconds to serve the HTML version of the page. Additionally, the status block tells us when the cached copy of the page is set to expire (based on our minimum cache lifetime settings).

You can also click on the **Flush Page** button and this will clear the Drupal page from the Boost cache entirely. Again, this is a very nice functionality to have if you are setting up a more complex Boost cache configuration.

Load some other pages both as an anonymous user and as your admin user, and check the status blocks for Status reports on the caching for that specific page.

**Boost: Pages cache configuration block**

This block allows you to configure your basic Boost settings for a specific Drupal page while you’re administering the page itself. So go ahead and launch a node on your site and you’ll see the block in the right sidebar. It will look similar to this:

![Boost Pages cache configuration block](image)

---

**For More Information:**
This block allows you to set and adjust the minimum cache lifetime (1 hour, 3 hours, and so on), the preemptive cache and whether it's enabled, as well as the scope of the Drupal page you are viewing. **Scope** will tell you the page ID of the node and what type of content the node is built from—in this case we're looking at a Photo content type node. The type of container (node, block, or otherwise) is also noted here. You can tweak the configuration settings here and save a new configuration. You can also delete your configuration here.

If you choose to set a new configuration here, for example, changing your page cache minimum cache lifetime to 3 hours versus 1 hour, the cache will be flushed and reset to your new 3 hour lifetime. So your status block will show that the page is being served live again on the site until you visit that page as an anonymous user. When the Drupal node is **being served live**, this is what the status block looks like:

![Status Block](image)

**Summary of Boost's basic configuration**

The one thing to remember when using the Boost module is that only your anonymous site visitors will be served static HTML page versions of your Drupal dynamic data. It will be business as usual for your authenticated site visitors. When they login, they will continue to receive the dynamic live version of the Drupal node. You will want to keep this in mind when you decide whether you will use this module on your site. If your site is heavily trafficked by anonymous users, then you'll want to try out Boost. If you have a community portal site that depends on authenticated user comments and forum activity, you may not want to use the Boost module or at least be aware of its limitations when it comes to how it works on that type of site.

Additionally, the Boost module only works on web servers running Apache server software. Much more about Boost is available on its Drupal module project page.

---

**For More Information:**

Summary
In this chapter, we looked in detail at two contributed modules—DB Maintenance and Boost, which help to both maintain and speed up performance of our website. Specifically, we did the following:

- Installed and configured the DB Maintenance contributed module in order to optimize, repair, and back up our MySQL database.
- Installed and configured the Boost module in order to set up basic static HTML page cached versions of our Drupal nodes.
- Tweaked our HTACCESS file to enable the Boost settings through the Boost htaccess rules generator functionality.
- Enabled the following Boost module blocks: Pages cache status, Pages cache configuration, and AJAX core configuration.

Let's take a break! When we come back in Chapter 6, we'll take a detailed look at Boost's advanced settings, including a discussion of how to troubleshoot the Boost module if it's not working correctly on your site.
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.