

Drupal 6 Performance Tips

Learn how to maximize and optimize your Drupal framework using Drupal 6 best practice performance solutions and tools

Trevor James

TJ Holowaychuk



BIRMINGHAM - MUMBAI

Drupal 6 Performance Tips

Copyright © 2010 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, Packt Publishing, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: February 2010

Production Reference: 1080210

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847195-84-5

www.packtpub.com

Cover Image by Nilesh Mohite (nilpreet2000@yahoo.co.in)

Credits

Authors

Trevor James
TJ Holowaychuk

Reviewers

Daniel Hanold
Joeri Poesen

Acquisition Editor

Douglas Paterson

Development Editor

Steven Wilding

Technical Editor

Akash Johari

Copy Editor

Lakshmi Menon

Indexer

Hemangini Bari

Editorial Team Leader

Akshara Aware

Project Team Leader

Lata Basantani

Project Coordinator

Joel Goveya

Proofreader

Kevin McGowan

Production Coordinator

Shantanu Zagade

Cover Work

Shantanu Zagade

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 (**V**irtual **T**raining **C**ompany) in 2009. The video is available via the VTC website here: <http://www.vtc.com/products/Introduction-To-Drupal-6-Tutorials.htm>

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>, 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.

About the Reviewers

Daniel Hanold is a software developer and business consultant based in New York City. After graduating from Stuttgart Media University with degrees in IT and Economics, he co-founded PeoplesMD, an online patient education resource. He began his love affair with Drupal shortly afterwards.

Equipped with his background in PHP and MySQL, Daniel creates applications ranging from brochure websites for high-profile clients to social networks for non-profit organizations. Daniel is an expert in combining CSS, JavaScript, and jQuery to make user interactions with any application as simple, elegant, and efficient as possible.

Currently, Daniel focuses on large scale community websites and performance optimization using technologies such as Apache Solr and Memcached. His personal blog can be found at <http://danielhanold.com>.

Joeri Poesen is a longtime Drupal user, developer, and trainer. He loves nothing more than scouring the planet, learning how open source tools such as Drupal empower individuals and organizations, and how he can contribute to their adoption.

When not traveling or organizing community events, Joeri is probably giving a Drupal training session somewhere in Europe—most likely in Paris, France.

I'd like to thank my wife Lies for her endless patience and support. We've been through some crazy things together the last 12 years, and I've got a feeling that we ain't seen nothin' yet.





*This book is dedicated to my parents, Michael and Judy James,
who taught me the value of hard work and dedication to the task at hand.*

– Trevor James



Table of Contents

Preface	1
Chapter 1: Upgrading Drupal	7
Upgrading Drupal 5.x core	8
Backing up your site and database	10
Taking your site offline	12
Running Status report	13
Upgrading to 5.19	15
Installing the Update Status module	16
Installing contributed module updates	19
Uninstalling and removing Update Status	20
Running cron and checking recent log entries	21
Dealing with contributed modules during upgrades	22
Backing up and exporting your Drupal 5.x Views	23
Reviewing your Panels code	26
Final prep for upgrading to 6.13	27
Disabling all contributed modules	28
Enabling the Garland theme site-wide	29
Downloading Drupal 6.13	29
Upgrading Drupal core	30
Running update.php	33
Upgrading contributed modules	38
Updating your PHP memory limit	41
Installing the updated Zen theme files	44
Upgrading your custom theme	44
Cleaning up and resetting Views	47
Placing your site back online	49
Summary	50

Table of Contents

Chapter 2: Maintaining your Drupal Site	51
Checking your Drupal configuration status	52
Checking your PHP and MySQL settings	55
Files to delete and clean up	57
Enabling the Update Status module	57
Disabling unused modules and themes	58
Introduction to Drupal caching	59
Enabling and configuring Drupal caching	61
Cache tables in your MySQL database	65
Clearing your performance cache	67
Clearing your theme registry	68
Running cron manually	69
Installing the Poormanscron module	70
Setting up cron through cPanel	73
Backing up your site using SFTP/FTP and cPanel	74
Backing up your database through phpMyAdmin	77
Tweaking your HTACCESS file	78
Summary	80
Chapter 3: Using Development Modules and Tools	81
Viewing and inspecting recent log entries	82
Viewing your recent log entries	82
Logging and alerts configuration	84
Page not found and access denied errors	84
The Devel module	86
Installing and enabling Devel	87
Checking Devel module permissions	88
Enabling Themer info	89
Devel settings	92
Inspecting database queries and Devel results	95
Enabling the Devel module block	98
Using the Devel module block	100
Database queries	100
Empty cache	101
Disable/Enable Theme developer	101
Execute PHP code	101
Function reference	103
Hook_elements()	103
PHPinfo()	104
Rebuild menus	104
Reinstall modules	105
Running cron	105
Session viewer	105

Theme registry	106
Variable editor	106
Summary	108
Chapter 4: Performance Optimization	109
Enabling and configuring the Throttle module	110
Configuring the Throttle module for auto throttling features	111
Throttling your modules	113
Throttling blocks	114
Generating test users, categories, and content	115
Views caching	120
Clearing your Views 2 module cache	124
Using Panels caching	127
Creating a panel and adding content to it	127
Summary	129
Chapter 5: Using DB Maintenance and Boost	131
Using the DB Maintenance module	132
Using the Boost module	135
Installing and configuring Boost	136
Boost settings	137
Boost File Cache settings	138
Boost cacheability settings	139
Boost directories and file extensions	140
HTACCESS file tweaks	142
Testing your Boost configuration	145
Boost and Poormanscron	147
Configuring Poormanscron	148
Clearing the Boost cache	148
Boost admin and stats blocks	148
Boost: Pages cache status block	149
Boost: Pages cache configuration block	150
Summary of Boost's basic configuration	151
Summary	152
Chapter 6: Advanced Boost	153
Updating contributed modules	154
Recommended modules that work with Boost	154
Global Redirect	155
Transliteration and Pathauto	156
Advanced Boost settings	157
Boost advanced settings	158
Testing your Database timestamp settings	162
Boost crawler settings	167
Summary	168

Chapter 7: Using Memcache API and Integration	169
Using the Memcache API and Integration module	170
MoWeS Portable development WAMP server	171
Installing Memcached libraries and service	172
Integrating and testing Memcached with PHP 5.2.x	173
Installing the Memcache API and Integration module	175
Enabling the Memcache Admin module	177
Memcache status	178
Memcache statistics per page	179
Viewing the Memcache tables in MySQL	181
Running Memcache without saving cache data to your database	181
Summary	182
Chapter 8: Advanced Caching and Contributed Modules for Caching	183
Cache Router	184
Cache Router versus Memcache API	185
Authenticated User Page Caching (Authcache)	186
Tweaking your settings.php file to support Authcache	187
Configuring the Authcache module	188
Page Caching Settings	190
Testing the Authcache module and its caching mechanism	192
Checking the AuthcacheFooter code	192
Checking the Authcache Debug window	192
Advanced cache	193
block_cache.patch	194
comment_cache.patch	194
forum_cache.patch	194
node_cache.patch	194
path_cache.patch	195
search_cache.patch	195
taxonomy_cache.patch	195
APC (Alternative PHP cache)	197
File Cache module	197
Summary	198

Chapter 9: Multisite Configuration and Performance	199
Using Drupal multisite	200
Configuring multisite in a localhost environment	201
Creating the multisite folders	202
Setting up databases for your multisite	202
Tweaking settings.php for each site	205
Editing your Apache configuration	206
Tweaking the hosts driver file on Windows	208
Tweaking the Base URL	209
Loading your multisites	209
Testing your multisite configuration	209
Using core and contributed modules in multisite	210
Installing modules and themes to a multisite	211
Setting themes per multisite	211
Caching and multisite	212
Enabling page caching and CSS/JS optimization per site	212
Multisite resources	213
Summary	214
Index	215

[Download at Wow! eBook](#)

Preface

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.

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.

What you need for this book

To follow along with the examples in this book, you will need a computer which can run MySQL, PHP, and the Apache web server, which are all prerequisites for Drupal. Luckily, every major operating system can run these applications. You may want to create an account with a website hosting company to test your work although you can also use a regular desktop or a laptop computer.

You will also need the Drupal Content Management framework, which is available from drupal.org. We will discuss downloading and installing Drupal in Chapter 1.

Who this book is for

This book is for Drupal website users and developers who want to boost and tweak performance on their website using Drupal's core and contributed performance module functionality.

You are expected to know about the basic operation of Drupal, be familiar with the concept of site configuration, and know how core and contributed modules are installed and work in Drupal. No experience with programming Drupal is required. The author will teach you how to implement specific code and patches that work with specific performance modules. Almost everything in the book will be focused on the module visual interface, and how to use this interface to configure and implement the performance module.

This book also covers upgrading Drupal, running Drupal security patches, creating backups of your Drupal site, and other basic Drupal site maintenance that will be helpful to the novice Drupal user and developer. Modules are covered in both their basic and advanced configurations; so both novice and advanced developer will learn how best to implement performance practices on their Drupal site with this step-by-step guide.

Performance is a large discussion within, and presents a large terrain to cover throughout, the Drupal community, but this book does not claim to cover every performance and site optimization issue. The authors have done their best to cover the majority of performance-based tips and tricks to run your Drupal site. They hope that the book will enhance the discussion of Drupal performance, and pave the way for more books and tutorials to be released on Drupal performance topics.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text are shown as follows: "Update the `$conf` in your `settings.php` file."

A block of code is set as follows:

```
<?php
$conf = array(
    // The path to wherever memcache.inc is. The easiest is to simply
    point it
    // to the copy in your module's directory.
    // 'cache_inc' => './sites/all/modules/memcache/memcache.inc',
    // or
    'cache_inc' => './sites/all/modules/memcache/memcache.db.inc',
);
?>
```

New terms and important words are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes, for example, appear in our text like this: "Once enabled, you can browse to this module's admin interface by going to **Reports | Memcache** status via your administrative menu."



Warnings or important notes appear in a box like this.



Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an e-mail to feedback@packtpub.com, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or e-mail suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.



Downloading the example code for the book

Visit http://www.packtpub.com/files/code/5845_CODE.zip to directly download the example code.

The downloadable files contain instructions on how to use them.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **let us know** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata are added to the list of existing errata. The existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide the location address or website name immediately so we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with some aspect of the book, and we will do our best to address it.



1

Upgrading Drupal

To run a high performance and secure Drupal website, you should keep your Drupal core code and your contributed Drupal module code patched and upgraded regularly. The Drupal project frequently releases updated security patches to its core code and it should be a standard maintenance workflow for you as a Drupal developer to keep your site patched to the latest core Drupal release. This will prevent security issues on your site (most of these patch releases are security upgrades) and it will help to improve your site's performance, as these patch releases fix minor and major issues reported from Drupal's bug tracking tools. Many performance issues and security issues with the Drupal code are fixed on a weekly to monthly basis by Drupal developers working with the Drupal project.

Besides security patches, Drupal releases major upgrades every 1-2 years. The most recent major upgrade was from Drupal 5.x to Drupal 6.x. Plans are underway now for the next major release, Drupal 7.x. It's good practice to begin the process of upgrading your Drupal 5.x site to Drupal 6.x as soon as possible, so you'll be in a better position to eventually upgrade and use Drupal 7.x. Major Drupal releases often solve performance issues reported over months and years by Drupal developers using Drupal sites. It's to your benefit and your site's future growth to upgrade soon after a major release appears.

All security patches and major Drupal upgrade releases are listed at the top of the home page at <http://drupal.org/> and each release provides background information on why the release has occurred.

Drupal 6.13 and 5.19 released

Drupal Security Team - July 1, 2009 - 20:18

[News and announcements](#) · [Drupal 5.x](#) · [Drupal 6.x](#) · [Drupal News](#)

Drupal 6.13 and 5.19, maintenance releases fixing problems reported using the bug tracking system, as well as **critical security vulnerabilities**, are now available for download. Both releases fix some other smaller issues as well.

[Download Drupal 6.13](#)
[Download Drupal 5.19](#)

Upgrading your existing Drupal 5 and 6 sites is strongly recommended. There are no new features in these releases. For more information about the Drupal 6.x release series, consult the [Drupal 6.0 release announcement](#), more information on the 5.x releases can be found in [Drupal 5.0 release announcement](#).

[» Read more](#)

In this chapter, you will learn how to maintain your Drupal site by doing the following:

- Upgrading your Drupal 5.x core to the latest 5.x version, and upgrading your Drupal 5.x contributed modules to the latest 5.x versions
- Backing up your entire Drupal site and database
- Running Status report
- Taking your Drupal site offline for maintenance
- Upgrading your Drupal 5.x site to Drupal 6.x
- Upgrading your contributed modules to the latest 6.x versions
- Running `update.php`
- Placing your new Drupal 6.x site back online

Learning these steps will allow you as a Drupal webmaster and developer to easily maintain your Drupal site and to troubleshoot performance issues with client websites. You will learn that immediately checking which version of Drupal and contributed modules the site is running can tell you a lot about the performance of the site.

There is a wealth of information about the upgrade process along with upgrade tutorials on drupal.org at the following URLs/pages:

- Upgrading from previous versions: <http://drupal.org/upgrade/>
- Upgrading from Drupal 5.x to Drupal 6.x: <http://drupal.org/videocasts/upgrading-to-6>

Upgrading Drupal 5.x core

Before we perform a major upgrade from Drupal 5.x to Drupal 6.x, we need to make sure our Drupal 5.x core and contributed modules are upgraded to the latest 5.x releases. For Drupal core, this is currently Drupal 5.19. We can determine the latest release by visiting <http://drupal.org/> and downloading the `.tar.gz` for this release. There are also release notes about each version located at <http://drupal.org/>. For example, the Drupal 5.x release notes are located on: <http://drupal.org/drupal-5.0>. You can read more release notes specific to a version number in the `CHANGELOG.txt` text file and the `UPGRADE.txt` files located in the Drupal root folder of that version.

Here is our 5.x upgrade plan:

We're going to upgrade a site running Drupal 5.18 in preparation for a full version upgrade to Drupal 6.x. The first thing we're going to do is to upgrade this site to Drupal 5.19. We're also going to upgrade all of our contributed modules to the latest 5.x versions of those modules. This is important to do before an upgrade to 6.x—if all of your modules and core code are the latest version of 5.x, it will make the upgrade process run more smoothly and leave less room for parse errors, white screens, and other upgrade issues.



Drupal.org notes the importance of upgrading to the latest minor version of your current Drupal version before starting an upgrade to a next major version release. So, in our case we need to upgrade to the latest version of Drupal 5.x (at the time of this writing, it is Drupal 5.19) before running an upgrade to Drupal 6.13. See the Drupal.org upgrade tutorial and articles at: <http://drupal.org/upgrade/>.

It may seem as though this workflow takes more time, but in the end your upgrade process will run more smoothly and with fewer problems.

The site we're upgrading is running Drupal 5.18. The theme is a custom version of the Zen StarterKit theme. The contributed modules on this site include CCK, Devel, Imagecache, Imagefield, FCK, Panels, Webform, jQuery, Views 1, and Lightbox. One of the main sections of the site includes an image gallery using Views and the Lightbox module to display photos of fire trucks. The Panels module is used to create a home page for the site displaying one blog post and the Lightbox-powered photo gallery.

Fire Truck Photos

[Antique Fire Apparatus Photos](#)

User login

Username: *

Password: *

[Create new account](#)

[Request new password](#)

Home

Zelus Quadrum

Submitted by admin on Thu, 07/23/2009 - 02:08.



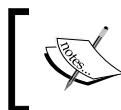
Backing up your site and database

Before beginning, run a full backup of both your Drupal directory and your Drupal database(s). If you run a backup, you can be confident that if an issue arises during the upgrade process causing possible corruption of your database, you can easily restore your site with your backup. The rule is to always run backups before performing any upgrades, even minor security patches.

If you are on a shared hosting server environment, you may have access to your site's directory and files through FTP/SFTP or a cPanel type of file manager utility (cPanel and Plesk being two of the common utilities). You can either use **file transfer protocol (FTP)** to download your directory to your local desktop or use a backup application that your host provides through your cPanel tool.

To backup your database, you can use a tool such as phpMyAdmin – this will allow you to connect to your MySQL database and export your full database to your local desktop (as a SQL file). Here are the steps to back up your site:

1. FTP/SFTP into your server and copy all of your Drupal files or directories to your local desktop.



A good habit is to name your local folder with the date of the backup: For example, `backup-09-08-15` – then if something happens, you can easily locate the last backup you ran.

Name	Ext	Size	Type	Changed	Attr	Owner
..			Parent directory	8/15/2009 ...		
files		8/15/2009 9:11...	rwxr-xr-x			variantc
includes		5/13/2009 2:01...	rwxr-xr-x			variantc
misc		7/25/2009 7:36...	rwxr-xr-x			variantc
modules		5/13/2009 2:01...	rwxr-xr-x			variantc
profiles		5/13/2009 2:01...	rwxr-xr-x			variantc
scripts		5/13/2009 2:01...	rwxr-xr-x			variantc
sites		5/13/2009 2:01...	rwxr-xr-x			variantc
themes		5/13/2009 2:01...	rwxr-xr-x			variantc
.htaccess		4,357 7/25/2009 8:00...	rw-r-r-			variantc
CHANGELOG.txt		33,500 5/13/2009 1:41...	rw-r-r-			variantc
cron.php		262 8/9/2006	rw-r-r-			variantc
error_log		1,505 7/27/2009 7:14...	rw-r-r-			variantc
index.php		872 12/12/2006	rw-r-r-			variantc
INSTALL.mysql.txt		1,431 9/8/2006	rw-r-r-			variantc
INSTALL.pgsql.txt		1,073 9/8/2006	rw-r-r-			variantc
install.php		22,305 7/9/2008	rw-r-r-			variantc
INSTALL.txt		9,263 1/10/2008	rw-r-r-			variantc
LICENSE.txt		18,049 1/13/2009	rw-r-r-			variantc
MAINTAINERS.txt		1,778 12/11/2006	rw-r-r-			variantc
robots.txt		1,591 12/10/2008	rw-r-r-			variantc
update.php		30,842 12/10/2008	rw-r-r-			variantc
UPGRADE.txt		2,941 1/9/2007	rw-r-r-			variantc
xmlrpc.php		352 12/10/2005	rw-r-r-			variantc

- Access your phpMyAdmin tool in your web browser (if your host provides access) and run an export of your Drupal database. Save the export file as a SQL (.sql) file in the same backup directory as your Drupal files.

The screenshot shows the phpMyAdmin interface for the 'variantc_drup5' database. On the left, a sidebar lists various tables: access, authmap, blocks, blocks_roles, book, boxes, cache, cache_content, cache_filter, cache_menu, cache_page, cache_views, comments, contact, content_type_blog, content_type_book, content_type_forum, content_type_page, content_type_photo, content_type_poll, content_type_story, content_type_webform, and devel_queries. The main panel displays the 'View dump (schema) of database' configuration. Under the 'Export' section, the 'access' table is selected. Below it, several export formats are listed: CSV, CSV for MS Excel, Microsoft Excel 2000, Microsoft Word 2000, LaTeX, and Open Document Spreadsheet. To the right, the 'Options' section includes fields for a custom comment, SQL compatibility mode (set to 'NONE'), and checkboxes for adding structure, IF NOT EXISTS, AUTO_INCREMENT values, backquotes, and CREATE PROCEDURE / FUNCTION. A 'Comments' section at the bottom contains a checkbox for creation/update/check dates.



3. If you are running your Drupal site locally (via localhost) or you have remote access to the actual server, you can simply copy the entire Drupal site folder and paste the folder into a backup folder on your server desktop or computer desktop. Then you can zip the backup or archive and save it until you know you have successfully completed the upgrade process. Before copying the entire Drupal site folder, you may want to check to make sure you can see all the files in your site folder. Occasionally, the operating system will hide files, for example your .htaccess file, so it's a good idea to show these files before backing up to make sure you are also backing up your .htaccess and any other hidden file.

Now you have a full backup of your Drupal site files and your database. You're ready to proceed with upgrading your site to 5.19.

Taking your site offline

The next step is to take your site offline for maintenance. It's good practice to always take your Drupal site offline during the upgrade process to prevent any public site visitor from accessing your site and viewing error messages that may appear during the upgrade. Your authenticated users with administrative level permissions will still be able to login to the site while it's offline. Good practice states that you should advise your authenticated user base that you will be performing site maintenance, so they avoid logging in during the process.

First login to your site as **user/1** (the super user admin) and then:

1. Click on the **Administer** link in your main navigation menu.
2. Click on the **Site Maintenance** link (in the **Site configuration** section of your admin screen).
3. Select the **Off-line** radio button.
4. Type in a Site offline message that your general public site visitors will see when they try launching your site while it's offline. You can add full HTML and images to this message.
5. Click on the **Save configuration** button.

Site maintenance

Site status:

Online
 Off-line

When set to "Online", all visitors will be able to browse your site normally. When set to "Off-line", only users with the "administer site configuration" permission will be able to access your site to perform maintenance; all other visitors will see the site off-line message configured below. Authorized users can log in during "Off-line" mode directly via the [user login](#) page.

Site off-line message:

Fire Truck Photos is currently under maintenance. We should be back shortly. Thank you for your patience.

Message to show visitors when the site is in off-line mode.

[Save configuration](#) [Reset to defaults](#)

Running Status report

You will be running the Status report often during the upgrade process to check on the status of your Drupal site. The Status report will tell you if you have any issues in your site that need to be resolved before trying an upgrade. To have upgrades run as smoothly as possible, it's good practice to run Status report and make sure the majority of your report is checked green telling you the site is operating smoothly. This will cut down on the amount of parse errors you get during the upgrade process as well as removing other upgrade problems. Let's go ahead and run it now to get a quick update on the site we'll be upgrading. The Status report tells us how our Drupal site is currently running and performing.

1. Go to your Drupal Logs' admin section and click on **Status report** (**Administer | Logs | Status report**)
2. Review the current status of your site. Notice that our site is running Drupal 5.18. This tells us immediately that we need to upgrade to Drupal 5.19 before doing a full upgrade to 6.x.

3. Notice that our PHP `memory_limit` is set too low for performance purposes. We're getting a flag with our **imagecache** settings due to the **32M** limit we have set currently. Once we complete the upgrade process, we'll want to return to this issue and raise our `memory_limit`. This is something we will cover in Chapter 2.
4. Run cron manually to flush the Status report and re-index our site. This also will allow you to check to see if any other red flags show up.

Status report

Here you can find a short overview of your Drupal site's parameters as well as any problems detected with your installation. It is useful to copy/paste this information when you need support.

Operating in off-line mode.

Drupal	5.18
✓ Configuration file	Protected
✓ Cron maintenance tasks	Last run 3 weeks 3 days ago You can run cron manually.
✓ Database schema	Up to date
✓ File system	Writable (<i>public</i> download method)
✓ GD library	bundled (2.0.34 compatible)
⚠ ImageCache PHP Memory Limit	32M It is highly recommended that you set your PHP <code>memory_limit</code> to 96M to use <code>imageapi_gd</code> . A 1600x1200 images consumes ~45M of memory when decompressed and ImageAPI is often operating on two decompressed images at once.

Status report is important when you're maintaining a client site for the first time. You can login to the site as admin and quickly get an update on the site and check the exact Drupal version the site is running. It also gives you a link to your PHP settings, which is something we will cover in Chapter 2.

Upgrading to 5.19

Let's go ahead and start the 5.19 upgrade. Here are the steps:

1. Download and extract the 5.19 tar.gz (from <http://drupal.org/>) to your local desktop. This will create a folder titled drupal-5.19.
2. Connect to your server through FTP/SFTP. Open up your local FTP window to show the Drupal 5.19 folder. Make sure your remote window shows your current Drupal site.
3. Select and move the following folders: includes, misc, modules, profiles, scripts, themes). Also, move over all of the root level files (cron.php, index.php, and so on). You do not need to move over the /sites folder as this has your customized settings.php file. You do not want to replace that file, so just ignore that folder. You do not want to replace your .htaccess file or your robots.txt file either, as those may also have customized code. The rest of the folders and files will only replace the core Drupal files and folders, updating their code with the latest version. You do not need to replace the /files folder either.
4. Start the transfer.



Name	Ext	Size	Type	Changed	Attr		Name	Ext	Size	Changed	Rights	Owner
..			Parent directory	8/16/2009			8/15/2009 9:11...	rwxr-xr-x	variantc
includes			Folder	8/16/2009 ...			files			5/13/2009 2:01...	rwxr-xr-x	variantc
misc			Folder	8/16/2009 ...			includes			7/25/2009 7:36...	rwxr-xr-x	variantc
modules			Folder	8/16/2009 ...			modules			5/13/2009 2:01...	rwxr-xr-x	variantc
profiles			Folder	8/16/2009 ...			profiles			5/13/2009 2:01...	rwxr-xr-x	variantc
scripts			Folder	8/16/2009 ...			scripts			5/13/2009 2:01...	rwxr-xr-x	variantc
sites			Folder	8/16/2009 ...			sites			5/13/2009 2:01...	rwxr-xr-x	variantc
themes			Folder	8/16/2009 ...			themes			5/13/2009 2:01...	rwxr-xr-x	variantc
.htaccess		4,242	HTACCESS File	2/26/2009 ...	a		.htaccess		4,357	7/25/2009 8:00...	rwxr-r--	variantc
CHANGELOG.txt		33,692	Text Document	7/1/2009 ...	a		CHANGELOG.txt		33,500	5/13/2009 1:41...	rwxr-r--	variantc
cron.php		262	PHP Script	8/9/2006 ...	a		cron.php		262	8/9/2006	rwxr-r--	variantc
index.php		872	PHP Script	12/12/2000 ...	a		error_log		1,505	7/27/2009 7:14...	rwxr-r--	variantc
INSTALL.mysql.txt		1,431	Text Document	9/8/2006 ...	a		index.php		872	12/12/2006	rwxr-r--	variantc
INSTALL.pgsql.txt		1,073	Text Document	9/8/2006 ...	a		INSTALL.mysql.txt		1,431	9/8/2006	rwxr-r--	variantc
install.php		22,305	PHP Script	7/9/2008 ...	a		INSTALL.pgsql.txt		1,073	9/8/2006	rwxr-r--	variantc
INSTALL.txt		9,263	Text Document	1/10/2008 ...	a		install.php		22,305	7/9/2008	rwxr-r--	variantc
LICENSE.txt		18,049	Text Document	1/14/2009 ...	a		INSTALL.txt		9,263	1/10/2008	rwxr-r--	variantc
MAINTAINERS.txt		1,778	Text Document	12/11/200...	a		LICENSE.txt		18,049	1/13/2009	rwxr-r--	variantc
robots.txt		1,591	Text Document	12/10/200...	a		MAINTAINERS.txt		1,778	12/11/2006	rwxr-r--	variantc
update.php		30,842	PHP Script	12/10/200...	a		robots.txt		1,591	12/10/2008	rwxr-r--	variantc
UPGRADE.txt		2,941	Text Document	1/9/2007 ...	a		update.php		30,842	12/10/2008	rwxr-r--	variantc
xmlrpc.php		352	PHP Script	12/10/200...	a		UPGRADE.txt		2,941	1/9/2007	rwxr-r--	variantc
							xmlrpc.php		352	12/10/2005	rwxr-r--	variantc

5. As soon as the files have transferred over through FTP, refresh your site's Status report. You should now see Drupal 5.19 listed.

6. Run cron manually again to see if you receive any new flag messages or errors. At this point Drupal will tell you that you need to update your database schema by running update.php. This is a crucial step during any Drupal upgrade process, as it updates your entire Drupal database schema. For minor version upgrades, you usually do not need to run update.php, but if in doubt, run your Status report and this will tell you whether you need to update your database schema.
7. If you are using the jQuery Update module in Drupal 5.x, you may receive a message in your Status report telling you to copy the jquery.js script file to the correct location. The upgrade process places this script into the /misc folder and will overwrite your previous jquery.js file. You will need to follow the instructions that Drupal gives you in your Status report and copy the file from sites/all/modules/jquery_update/misc/jquery.js.



Please copy jQuery

Copy jquery.js

In order for the jQuery Update module to work correctly, please copy the file at *sites/all/modules/jquery_update/misc/jquery.js* and use it to replace *misc/jquery.js*.

8. Run cron once more to check Status report. You should now have a fully upgraded core Drupal 5.19 site.

Installing the Update Status module

The next step is to upgrade any contributed modules to the latest 5.x versions. This is a requirement before trying to upgrade to version 6. To determine if there's a new version of our contributed modules, we can install the Drupal Update Status module. Drupal 6.x has update status as part of its core modules, but with Drupal 5.x you need to install a contributed module. This will give you some practice in installing Drupal modules.

1. Go directly here: http://drupal.org/project/update_status and download the 5.x-2.3 version.

Update Status

[View](#) [CVS instructions](#) [Revisions](#)

dww - February 25, 2007 - 21:55

Administration · Modules

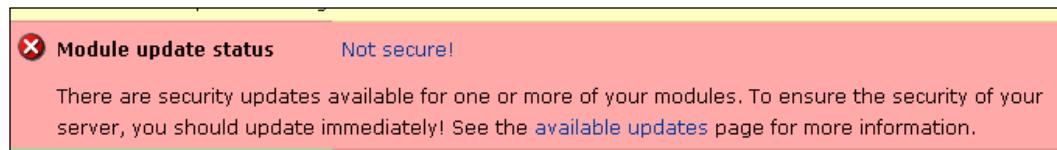
Module	Version	Status	Actions
Drupal 5.6	5.7 (2008-Jan-29)	Update available	Download · Release notes
cck.pre-rename 5.x-1.x-dev	(2008-Jun-01)	No available releases found	
Coder 5.x-2.x-dev	(2008-Jun-01)	Up to date	Download · Release notes

The Update Status module checks with drupal.org once a day to see if there are new officially released versions of Drupal and any modules that you are running. It requires cron to do its thing, so make sure that you have cron correctly enabled or it won't be able to know.

2. Extract the `.tar.gz` file to your desktop and then copy the `update_status` folder into your `/sites/all/modules` folder using FTP.
3. Enable the module in your modules admin at **Administer | Site building | Modules**. Save your module configuration.

Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	FCKeditor	5.x-2.2	Enables the usage of FCKeditor (WYSIWYG) instead of plain text fields.
<input checked="" type="checkbox"/>	IMCE	5.x-1.2	An image/file uploader and browser supporting personal directories and user quota.
<input checked="" type="checkbox"/>	Panels	5.x-1.2	Create pages that are divided into areas of the page. Required by: Bonus: Panels (enabled)
	<input checked="" type="checkbox"/> Update status	5.x-2.3	Checks to see if your installation of Drupal and its modules are up to date.
<input checked="" type="checkbox"/>	Webform	5.x-2.5	Enables the creation of forms and questionnaires.

4. Go back to your Status report. You should see a red-flagged box notifying you that your modules need updating. The **Module update status** box will state: **Not secure!**



5. Click on the **available updates** link to see what contributed modules in your site need updating.
6. The Available Updates page will load and all modules needing updates will be flagged in red with the recommended version listed. The module provides a link to the module's project page at <http://drupal.org/>, so you can easily download the most recent version to your desktop.
7. Notice in our list, we need to update the **ImageField** and the **Lightbox2** modules.

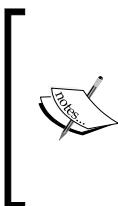
The screenshot displays a list of contributed modules and their update status. The modules are listed in rows, each with a green header and a pink body. The columns from left to right are: module name, recommended version, update availability, and download/release notes.

Module	Recommended version	Update available	Action
ImageField 5.x-2.5	5.x-2.6 (2009-Mar-16)	✖	Download · Release notes
IMCE 5.x-1.2		Up to date ✓	
jQuery plugins 5.x-1.3		Up to date ✓	
jQuery Update 5.x-2.0		Up to date ✓	
Lightbox2 5.x-2.8	5.x-2.9 (2009-Jan-08)	✖	Download · Release notes

Installing contributed module updates

Remember that you need to update your contributed modules to the latest 5.x released versions before attempting to upgrade your Drupal site to 6.x. Before upgrading the 5.x versions of contributed modules, it's a good idea to back up your site files and database once more now that you're at the core Drupal 5.19 release.

1. Download the latest stable release of the **Imagefield**, **Lightbox2**, and **Webform** modules. The Project pages are here:
 - <http://drupal.org/project/imagefield>
 - <http://drupal.org/project/lightbox2>
 - <http://drupal.org/project/webform>
2. Copy the extracted folders to your `/sites/all/modules` folder to replace current versions. Be aware here that you will be overwriting your contributed module code. So if you have made any hacks or edits to contributed code, you will need to replace those files later after you've upgraded.



Remember that the golden rule followed by the Drupal project and community is *do not hack core*. This is also a good rule to follow with your contributed modules. If you have to manipulate module code, do this by using proper Drupal theme and module overrides in your `template.php` file.

3. Once you have replaced your **Imagefield**, **Lightbox2**, and **Webform** modules, refresh your available updates page.
4. Notice that in our FTP screenshot there's a bunch of `tar.gz` folders residing in our `/sites/all/modules` directory. Sometimes you'll find these zipped folders remaining in your modules directory. They are just taking up unnecessary space. You can delete them safely from your `/sites/all/modules` directory.
5. Once you copy over your new versions, your available updates page should be green next to each module.
6. Run your Status report again—this is important because Drupal will then let you know if you need to run `update.php`. Very often with contributed module upgrades and updates, you need to update your database schema. It's recommended to run `update.php` after every contributed module upgrade.

7. In this case, after running Status report we do not need to update our database schema. The three modules we updated did not make any database level changes. So, we're good to go with our Drupal 5.19 upgrade. The update/upgrade process is now complete and we're ready to move onto the Drupal 6.x upgrade procedure.

Uninstalling and removing Update Status

Before proceeding with the full version upgrade to Drupal 6.x, we must uninstall and remove the Drupal Update Status module. This module comes packaged with Drupal 6.x core, so we cannot leave the 5.x version of the **Update status** module in place.

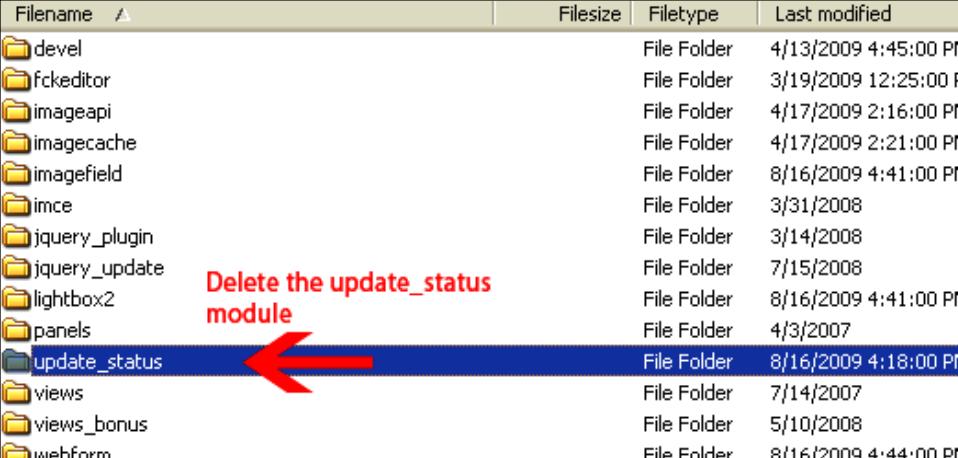
1. Go to your modules admin page (**Administer | Site Building | Modules**).
2. Uncheck the **Update status** module checkbox and save the module configuration. This will disable the module.
3. Click on the **Uninstall** tab at the top of your modules page. This will load the uninstall list and any modules available (disabled modules) for uninstalling.
4. Check the **Update status** module box.
5. Click on the **Uninstall** button.

The screenshot shows the Drupal 6.x Modules page. The URL in the address bar is "Home > Administer > Site building". Below the address bar, there are tabs: "Modules", "List", and "Uninstall" (which is highlighted). A message below the tabs says: "The uninstall process removes all data related to a module. To uninstall a module, you must first disable it. Not all modules support this feature." A green box contains the text "Operating in off-line mode.". The main table has columns: "Uninstall", "Name", and "Description". There is one row: "Uninstall" (with a checked checkbox), "Update status", and "Checks to see if your installation of Drupal and its modules are up to date.". At the bottom left is a red box around the "Uninstall" button, and a red arrow points from the text above to this button.

Uninstall	Name	Description
<input checked="" type="checkbox"/>	Update status	Checks to see if your installation of Drupal and its modules are up to date.

6. Drupal will ask you to confirm that you want to uninstall the module. Click on **Uninstall** again.
7. The module will be uninstalled.

8. If you revisit your module admin list, you'll notice that the **Update status** module is still listed, but is not checked/enabled. You need to also delete the entire file folder for this module from your `/sites/all/modules` directory before proceeding with the upgrade. If any files are lingering on the server for this module, you could generate errors during the upgrade process.
9. FTP into your server and remove the `update_status` folder from `/sites/all/modules`.



Filename	A	Filesize	Filetype	Last modified
devel			File Folder	4/13/2009 4:45:00 PM
fckeditor			File Folder	3/19/2009 12:25:00 PM
imageapi			File Folder	4/17/2009 2:16:00 PM
imagecache			File Folder	4/17/2009 2:21:00 PM
imagefield			File Folder	8/16/2009 4:41:00 PM
imce			File Folder	3/31/2008
jquery_plugin			File Folder	3/14/2008
jquery_update			File Folder	7/15/2008
lightbox2			File Folder	8/16/2009 4:41:00 PM
panels			File Folder	4/3/2007
update_status			File Folder	8/16/2009 4:18:00 PM
views			File Folder	7/14/2007
views_bonus			File Folder	5/10/2008
webform			File Folder	8/16/2009 4:44:00 PM

10. Refresh your modules admin list and the `Update_status` module table row should be deleted.

Running cron and checking recent log entries

Before proceeding with the upgrade to Drupal 6.x, complete the following steps:

1. Run cron again here: `/admin/logs/status`.
2. View recent log entries to make sure you're not getting any parse errors or other errors in your current Drupal 5.19 site. It's a good idea to check and try to get as few errors as possible in your recent log reports before upgrading to Drupal 6.x. This will make the process go smoother.

3. To check recent log entries, go to **Administer | Logs | Recent log entries**.

Your report or table should list all blue links and successful tasks. If you notice any red errors, make a note of them to troubleshoot before going forward. You should see something similar to the following screenshot (notice that no errors are visible):

The screenshot shows a table titled 'Recent log entries' with the following columns: Type, Date, Message, User, and Operations. The table lists various log entries from August 16, 2009, to August 17, 2009. Most entries are 'cron' or 'update_status' type, indicating successful cron runs and module updates. A few 'user' entries show session openings and closings. All entries are listed in blue, indicating they are successful. The 'Operations' column shows 'view' for some entries.

Type	Date	Message	User	Operations
cron	08/17/2009 - 20:32	Cron run completed.	admin	
user	08/17/2009 - 20:24	Session opened for admin.	admin	
user	08/16/2009 - 21:03	Session closed for admin.	admin	
cron	08/16/2009 - 21:03	Cron run completed.	admin	
update_status	08/16/2009 - 20:53	Fetched information about all available new releases ...	admin	view
cron	08/16/2009 - 20:49	Cron run completed.	admin	
update_status	08/16/2009 - 20:44	Fetched information about all available new releases ...	admin	view
update_status	08/16/2009 - 20:42	Fetched information about all available new releases ...	admin	view
update_status	08/16/2009 - 20:21	Fetched information about all available new releases ...	admin	view
cron	08/16/2009 - 20:00	Cron run completed.	admin	
cron	08/16/2009 - 19:54	Cron run completed.	admin	
cron	08/16/2009 - 19:21	Cron run completed.	admin	
user	08/16/2009 - 18:54	Session opened for admin.	admin	
user	08/16/2009 - 04:29	Session closed for admin.	admin	

Dealing with contributed modules during upgrades

In our upgrade process to Drupal 6.x, I'm going to focus on two contributed modules and the process of upgrading these two modules (as well as our other contributed modules) to their 6.x versions. When upgrading your contributed modules, it's good practice to read all of the module project information on the drupal.org website. Visit the Drupal contributed module project page for details about the module and how the module behaves during the upgrade process. You will also want to confirm that your contributed modules have 6.x versions. If the module has not been ported to 6.x yet, you will not want to proceed with an upgrade to Drupal 6.x unless you can live without that module's functionality on your website. Many of the contributed modules have been ported to Drupal 6.x at this time, but some are still in development mode. You are responsible for checking for this information and confirming the release version before upgrading.

While viewing a module's project page at drupal.org, look for the release versions in the **Releases** table at the bottom of the project page. For example, for the **Views** module you will see this:

Releases					
Official releases	Date	Size	Links	Status	
6.x-2.6	2009-Jun-10	1.46 MB	Download · Release notes	Recommended for 6.x	✓
5.x-1.6	2007-Jul-14	207.36 KB	Download · Release notes	Recommended for 5.x	✓
Development snapshots					
6.x-2.x-dev		2009-Aug-12	1.47 MB	Download · Release notes	Development snapshot ✗
5.x-1.x-dev		2009-May-06	243.74 KB	Download · Release notes	Development snapshot ✗
View all releases 					

Official releases with a green background and blue hyperlinks mean that the release is stable and ported for that version of Drupal. So, Views has a valid and stable 6.x release according to the project page. You will also see many development versions (snapshots) of the release. Use stable versions as much as possible. Development versions should not be run on production web servers.

Backing up and exporting your Drupal 5.x Views

Drupal 5.x uses Views version 1.x. Our site is currently using the last version of Views for Drupal 5.x (Views 5.x-1.6). The Views module is notoriously difficult to upgrade to the latest version of Views2 for Drupal 6.x. You will want to make sure you export all of your View code from the Drupal 5.x version, and keep it in backup notepad files in case you need this code to rebuild your Views in Views2 in the event you receive errors during the upgrade process. This is a good practice before upgrading the Views module.

On our site we have two custom Views built, both to control photo galleries on the site. We have a general Photo Gallery view using the Lightbox2 module plugin, and we have a new View that the developer of the site is working on to integrate some jQuery code for the display of the images. This view is not completed and enabled yet, but it's a good idea to export the code to have a backup if necessary.

This is also a great time in the upgrade process to see if you have Views in your site that you no longer need and can delete. Cleaning house is good practice before an upgrade to 6.x. You can export the View code for your disabled Views and keep a backup of that exported code. Then, you can delete those Views and not have to worry about converting them in the upgrade to 6.x.

Default Drupal Views that come packaged in the Drupal Views module should convert and translate without issues to Views2, but if in doubt it's a good idea to export these Views as well.

For more on the Views module visit: <http://drupal.org/project/views>. For Views2 documentation, you'll need to install the Advanced Help module (we'll do this after we upgrade to Drupal 6.x.).

1. To export your Views, click on the **Export** link under the **Actions** column in your Views list, next to the View you want to export.

Home > Administer > Site building

Administer views [List](#) [Add](#) [Import](#) [Theme wizard](#) [Tools](#)

This screen shows all of the views that are currently defined in your system. The default views are provided by Views and other modules and are automatically available. If a customized view of the same name exists, it will be used in place of a default view.
[more help...]

Operating in off-line mode.

Existing Views						
View	Title	Description	Provides	URL	Actions	
photo_gallery	Antique Fire Apparatus & Muster, Harrisburg, PA	Photo Gallery	Page	photo_gallery	Edit	Export
photo_gallery_jquery	Antique Fire Apparatus & Muster, Harrisburg, PA	Photo Gallery JQuery Test	Page, Block	photo_gallery_jquery	Edit	Export

Below are system default views; if you edit one of these, a view will be created that will override any system use of the view.



2. This will launch a screen that contains all of your View's export code. Right-click on the code window and select all, or copy all of the code using your mouse to select it.

Home > Administer > Site building > Views

View View Edit Clone Export

Operating in off-line mode.

```
$view = new stdClass();
$view->name = 'photo_gallery';
$view->description = 'Photo Gallery';
$view->access = array (
  '1',
);
$view->view_args_php = '';
$view->page = TRUE;
$view->page_title = 'Antique Fire Apparatus & Muster, Harrisburg, PA';
$view->page_header = '';
$view->page_header_format = '1';
$view->page_footer = '';
$view->page_footer_format = '1';
$view->page_empty = '';
$view->page_empty_format = '1';
$view->page_type = 'bonus_grid';
$view->url = 'photo_gallery';
$view->use_pager = TRUE;
$view->nodes_per_page = '24';
$view->sort = array (
  array (
    'tablename' => 'node',
    'field' => 'created',
    'sortorder' => 'ASC',
    'options' => 'normal',
  ),
);
$view->argument = array (
);
$view->field = array (
  array (
    'tablename' => 'node_data_field_photo_gallery_photo',
    'field' => 'field_photo_gallery_photo_fid',
    'label' => '',
    'handler' => 'content.views.field.handler_group'
  )
);
```

Click Export and select all

3. Paste the exported code into WordPad or Notepad and save the file with the following naming convention—use the name of your `View_code.txt`. So, for our example we'll name the file `photo_gallery.txt`. This can be a text file, as you'll be copying and pasting the code directly from this file back into Drupal later in the 6.x version, if you need it while using the Views Import Code utility.
4. Do the same process for your other Views if you have multiple Views.
5. You now have export backups for all your View code.

Reviewing your Panels code

The other module that tends to give developers a difficult time during the upgrade process is Panels. We'll be upgrading from Panels 5.x-1.2 to the latest stable version for Drupal 6.x, Panels3 (6.x-3.0-rc1). Upgrading to and installing Panels 3 will also require us to add another dependent module called CTools. You can learn more about Panels3 and CTools here: <http://drupal.org/project/panels>.

It's a good idea to make notes about your panel configuration—the type of panel you created originally in Drupal 5.x, the content you added to the Panel, and any other detailed configuration you may have added, including CSS styles, and so on. You may need this information to rebuild your Panels in Drupal 6.x if you run into issues or errors.

So, on our site we have one Panel titled **Home** with a URL/path of **home**. If we click on the **Edit** link in our Panels admin, we can view details about the panel.

The screenshot shows the 'Panels' administration page. At the top, there are tabs for 'Panels', 'List', and 'Add'. Below the tabs, a message says 'You may peruse a list of your current panels layouts and edit them, or click add to create a new page.' A note below that says 'Operating in off-line mode.' The main table lists one panel:

Panel title	URL	Operations
Home	home	 Edit Delete

In our panel we do not have any CSS specified. The panel is a two column stacked panel. We do have two pieces of content loaded into our panel layout. Our top pane has a node and a View loaded. Our other panes are empty. We'll make a note of this layout arrangement in a WordPad document and save it to our desktop for reference later.

The screenshot shows the 'General information' section of a panel configuration. It includes fields for Page title (Home), CSS ID (empty), Access (checkboxes for anonymous and authenticated users), Path (home), and a preview icon showing a two-column layout.

The 'Top' pane contains a node (Zelus Quadrum) and a view (photo_gallery). The 'Left side' and 'Right side' panes both say 'This area has no content.' The 'Bottom' pane also says 'This area has no content.'

Final prep for upgrading to 6.13

Here's a final prep checklist before we start the 6.13 upgrade:

1. Run cron manually a final time before the upgrade.
2. Check your recent log entries one final time at **Administer | Logs | Recent log entries** before upgrading.

3. Make a list of all of your contributed modules so that you have a module checklist to work from during the upgrade process. Here is our list of contributed modules for the site that we'll need to update once we upgrade Drupal core:

- Administration Menu
- CCK
- Development (Devel)
- Imagecache
- FCKEditor
- IMCE
- Lightbox2
- Panels
- Webform
- jQuery plugins
- Views

It's also good practice to print out the list of your modules before doing the upgrade, so that you have the printout as a reference to fall back on when you're downloading your 6.x versions.

Disabling all contributed modules

Access your Modules list and uncheck all contributed modules to disable them in advance of the upgrade. You do not need to uninstall any of the modules (apart from the Drupal Update Status module that we already removed), but it's a good idea to disable all contributed modules.

Administration				
Enabled	Name	Version	Description	Operations
<input checked="" type="checkbox"/>	Administration Menu	5.x-2.8	Renders a menu tree for administrative purposes as dropdown menu at the top of the page.	Edit Uninstall

Save your module configuration once you have disabled all the contributed modules.

Enabling the Garland theme site-wide

Before upgrading it's a good idea to disable your custom theme (in this case we are using a sub-theme of Zen called 'Apollo'). Enable the Garland default Drupal theme (the one you're currently using on the admin portions of the site) for your entire site during the upgrade process. We're going to be deleting the entire Zen theme in order to upgrade it to the 6.x version of Zen, so let's disable it first.

1. Go to the **Administer | Site building | Themes** admin list and disable the **Apollo** theme.
2. Check the **Default** radio button next to the **Garland** theme to make it your default system theme for the upgrade process.

Screenshot	Name	Enabled	Default	Operations
	STARTERKIT: sites/all/themes/zen/STARTERKIT	<input type="checkbox"/>	<input type="radio"/>	
	apollo: sites/all/themes/zen/apollo	<input type="checkbox"/>	<input type="radio"/>	configure
	bluemarine: themes/bluemarine	<input type="checkbox"/>	<input type="radio"/>	
	chameleon: themes/chameleon	<input checked="" type="checkbox"/>	<input type="radio"/>	configure
	garland: themes/garland	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>	configure

3. Save configuration.

Downloading Drupal 6.13

Download the latest stable release of Drupal (for this book it's 6.13) from the drupal.org home page.

1. Click on **Download Drupal 6.13** and then save the tar.gz file to your local desktop.

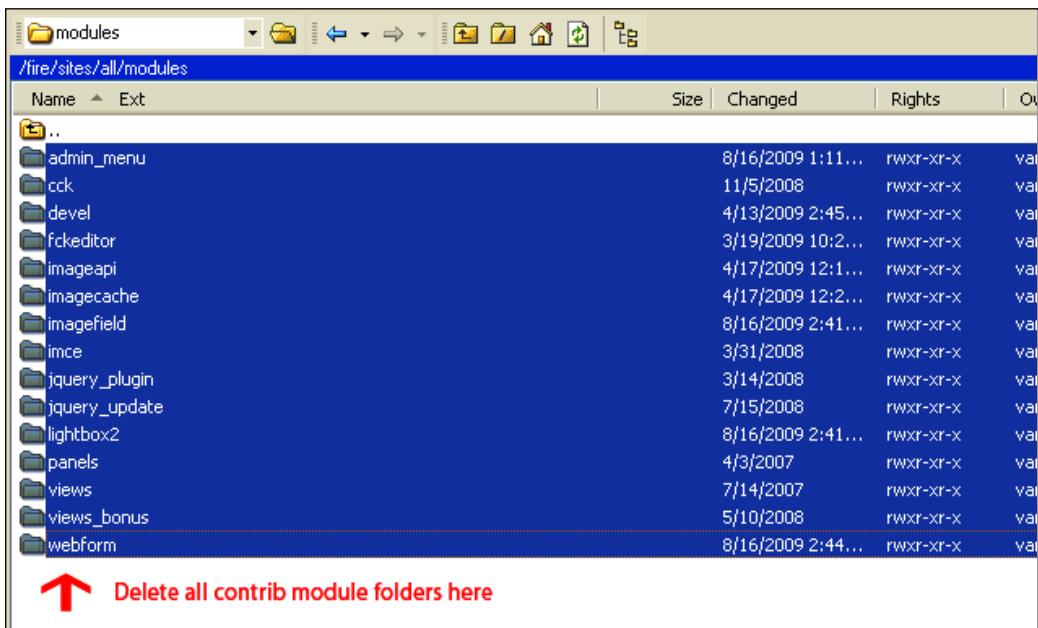


2. Extract tar.gz to your desktop. This will create a drupal-6.13 folder.

Upgrading Drupal core

We're going to upgrade the Drupal core code first. Here are the steps:

1. Login to your server through FTP/SFTP.
2. You should have your local drupal-6.13 folder on your local FTP side (left side) and your Drupal site on your right side in the FTP window.
3. Open up /sites/all/modules and delete your entire contributed module list—you can safely delete all the contributed module code as you'll be adding the new Drupal 6.x versions during the upgrade process. It's a good idea to remove the legacy module folders first.



4. After backing up your custom theme files (make sure you have a good backup of all your CSS, and so on), delete the entire custom theme folder (in this case zen). We'll add our custom theme CSS to the site once we complete the core upgrade process.



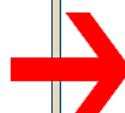
5. Delete all of your core Drupal files taking care not to delete the /sites or /files folders. Do not delete your .htaccess, robots.txt files, or any other files you may have added custom code to either. This includes php.ini. These files can remain in your Drupal directory.

Name	Ext	Size	Changed	Rights	Own
...					
files			8/15/2009 9:11...	rwxr-xr-x	var
includes			8/16/2009 1:35...	rwxr-xr-x	var
misc			8/16/2009 2:00...	rwxr-xr-x	var
modules			5/13/2009 2:01...	rwxr-xr-x	var
profiles			5/13/2009 2:01...	rwxr-xr-x	var
scripts			8/16/2009 1:37...	rwxr-xr-x	var
sites			5/13/2009 2:01...	rwxr-xr-x	var
themes			5/13/2009 2:01...	rwxr-xr-x	var
.htaccess		4,357	7/25/2009 8:00...	rw-r--r--	var
CHANGELOG.txt		33,692	8/16/2009 1:38...	rw-r--r--	var
cron.php		262	8/16/2009 1:38...	rw-r--r--	var
error_log		1,505	7/27/2009 7:14...	rw-r--r--	var
index.php		872	8/16/2009 1:38...	rw-r--r--	var
INSTALL.mysql.txt		1,431	8/16/2009 1:38...	rw-r--r--	var
INSTALL.pgsql.txt		1,073	8/16/2009 1:38...	rw-r--r--	var
install.php		22,305	8/16/2009 1:38...	rw-r--r--	var
INSTALL.txt		9,263	8/16/2009 1:38...	rw-r--r--	var
LICENSE.txt		18,049	8/16/2009 1:38...	rw-r--r--	var
MAINTAINERS.txt		1,778	8/16/2009 1:38...	rw-r--r--	var
robots.txt		1,591	12/10/2008	rw-r--r--	var
update.php		30,842	8/16/2009 1:38...	rw-r--r--	var
UPGRADE.txt		2,941	8/16/2009 1:38...	rw-r--r--	var
xmlrpc.php		352	8/16/2009 1:38...	rw-r--r--	var



Delete all core Drupal folders/files

6. Move your Drupal 6.13 files from your local side to your remote server – this will add all of the Drupal 6.13 core folders and files to your Drupal site/server. You do not need to move over the /sites folder, .htaccess, or robots.txt files. If you accidentally do move over the /sites folder, you will not replace your settings.php file because Drupal 6.x names this file with a different file name, so you cannot accidentally overwrite the file during an upgrade.



Name	Ext	Size	Type	Changed	Attr
..			Parent directory	8/18/2009 10:36:53 AM	
themes			File Folder	8/18/2009 10:36:53 AM	
sites			File Folder	8/18/2009 10:36:53 AM	
scripts			File Folder	8/18/2009 10:36:53 AM	
profiles			File Folder	8/18/2009 10:36:53 AM	
modules			File Folder	8/18/2009 10:36:53 AM	
misc			File Folder	8/18/2009 10:36:51 AM	
includes			File Folder	8/18/2009 10:36:51 AM	
xmlrpc.php		352	PHP Script	12/10/2005 3:26:48 PM	a
UPGRADE.txt		5,002	Text Document	1/4/2008 12:15:58 PM	a
update.php		25,457	PHP Script	3/30/2009 7:15:12 AM	a
robots.txt		1,590	Text Document	12/10/2008 4:12:20 PM	a
MAINTAINERS.txt		1,924	Text Document	4/29/2009 1:15:10 PM	a
LICENSE.txt		18,048	Text Document	1/6/2009 1:27:14 PM	a
INSTALL.txt		15,646	Text Document	7/9/2008 3:16:00 PM	a
install.php		46,926	PHP Script	4/27/2009 6:50:36 AM	a
INSTALL.pgsql.txt		1,075	Text Document	11/26/2007 12:36:42 PM	a
INSTALL.mysql.txt		1,308	Text Document	11/19/2007 3:53:52 PM	a
index.php		980	PHP Script	12/26/2007 4:46:48 AM	a
cron.php		262	PHP Script	8/9/2006 3:42:56 AM	a
COPYRIGHT.txt		981	Text Document	2/6/2008 8:45:56 AM	a
CHANGELOG.txt		42,420	Text Document	7/1/2009 4:51:56 PM	a
.htaccess		3,837	HTACCESS File	12/10/2008 4:04:08 PM	a

Name	Ext
..	
files	
sites	
.htaccess	
robots.txt	

7. Refresh your Drupal site (on whatever admin page you are currently on). If you receive immediate parse errors or a white screen (the infamous white screen of death, though it's really nothing to be afraid of), make sure to carry out the following steps and run your `update.php` to update the database schema. You should see a screen with a bunch of parse errors at this point:

Running update.php

Immediately after moving your new Drupal 6.x core files over to your server and refreshing your site page, you'll see a bunch of parse errors. Don't panic! Follow these steps to run your update.php script in order to update your Drupal database schema, so it updates to the latest Drupal 6.x configuration:

1. Type in update.php at the root level of your site URL. So, for this site we'll have the following URL to run our script: <http://variantcube.com/fire/update.php>
 2. Click the Return button on your keyboard to load the update.php script.

3. You will see the following Drupal database update page (along with more parse errors):

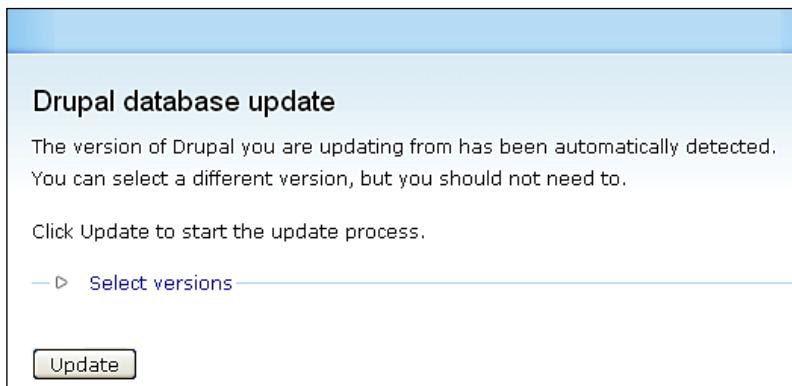
The screenshot shows a Drupal administrative interface titled "Fire Truck Photos". On the left, there's a sidebar with a logo and links: "Overview" (which is expanded), "Select updates", "Run updates", and "Review log". The main content area is titled "Drupal database update". It contains a red box listing three user warnings about non-existent tables:

- user warning: Table 'variantc_drup5.menu_router' doesn't exist query:
SELECT * FROM menu_router WHERE path IN ('admin/build/themes','admin /build/%','admin/%/themes','admin/build','admin/%','admin') ORDER BY fit DESC LIMIT 0, 1 in /home/variantc/public_html/fire/includes/menu.inc on line 315.
- user warning: Table 'variantc_drup5.menu_router' doesn't exist query:
SELECT * FROM menu_router WHERE path IN ('admin/build/themes','admin /build/%','admin/%/themes','admin/build','admin/%','admin') ORDER BY fit DESC LIMIT 0, 1 in /home/variantc/public_html/fire/includes/menu.inc on line 315.
- user warning: Table 'variantc_drup5.menu_router' doesn't exist query:
SELECT * FROM menu_router WHERE path IN ('admin/build/themes','admin /build/%','admin/%/themes','admin/build','admin/%','admin') ORDER BY fit DESC LIMIT 0, 1 in /home/variantc/public_html/fire/includes/menu.inc on line 315.

Below the error list, there's a note: "Use this utility to update your database whenever a new release of Drupal or a module is installed." and a link to the "Installation and upgrading handbook". A numbered list of steps for upgrading follows:

1. **Back up your database.** This process will change your database values and in case of emergency you may need to revert to a backup.
2. **Back up your code.** Hint: when backing up module code, do not leave that backup in the 'modules' or 'sites/*/modules' directories as this may confuse Drupal's auto-discovery mechanism.
3. Put your site into [maintenance mode](#).
4. Install your new files in the appropriate location, as described in the handbook.

4. Click on the **Continue** button to begin the update process.
5. A second Drupal database update page will load. You do not need to select the versions—Drupal will do this automatically. Click on the **Update** button.



6. The updates will run. You will see a progress bar and each core module will be updated in the database schema. Allow all the database updates to complete. Do not click your mouse during this process.
7. Once completed, you should see a final **Drupal database update** page load. A series of messages will be visible telling you the database update is complete. For example, on our site Drupal tells us that it now has **separate edit and delete permissions**. This is good information to read, as you know exactly what Drupal did during the upgrade process. You'll also see a long list of executed database queries. This list will tell you exactly what updates the Drupal update.php script performed in your database tables. For example, if the update script altered or replaced tables.

8. At this point your update page should show all green messages and gray query executed updates. You should also see green checks next to your upgrade list at the top left corner of the page – **Overview**, **Select Updates**, **Run updates**, and **Review log**.

<ul style="list-style-type: none"><input checked="" type="checkbox"/> Overview<input checked="" type="checkbox"/> Select updates<input checked="" type="checkbox"/> Run updates<input checked="" type="checkbox"/> Review log	<p>Drupal database update</p> <ul style="list-style-type: none">○ Drupal can check periodically for important bug fixes and security releases using the new update status module. This module can be turned on from the modules administration page. For more information please read the Update status handbook page.○ Drupal now has separate edit and delete permissions. Previously, users who were able to edit content were automatically allowed to delete it. For added security, delete permissions for individual core content types have been removed from all roles on your site (only roles with the "administer nodes" permission can now delete these types of content). If you would like to reenable any individual delete permissions, you can do this at the permissions page.○ User signatures no longer inherit comment input formats. Each user's signature now has its own associated format that can be selected on the user's account page. Existing signatures have been set to your site's default input format. <p>Updates were attempted. If you see no failures below, you may proceed happily to the administration pages. Otherwise, you may need to update your database manually. All errors have been logged.</p> <ul style="list-style-type: none">○ Main page○ Administration pages <p>The following queries were executed</p> <p>system module</p> <p>Update #6000</p> <ul style="list-style-type: none">○ ALTER TABLE {boxes} CHANGE COLUMN bid bid int NOT NULL○ REPLACE INTO {sequences} VALUES ('{boxes}_bid', 0) <p>Update #6001</p> <ul style="list-style-type: none">○ ALTER TABLE {term_node} ADD `vid` INT unsigned NOT NULL DEFAULT 0○ ALTER TABLE {term_node} DROP PRIMARY KEY○ ALTER TABLE {term_node} ADD PRIMARY KEY (vid, tid, nid)○ ALTER TABLE {term_node} ADD INDEX vid (vid)	
--	--	--

9. Your core upgrade process is now completed from the database side.
10. Click on the **Main page** or **Administration pages** link to be taken back to your new 6.13 site. It's a good idea to first click on the **Administration pages** link to make sure the admin pages load correctly.

Updates were attempted. If you see no failures below, you may proceed happily to the [administration pages](#). Otherwise, you may need to update your database manually. All errors have been [logged](#).

- [Main page](#)
- [Administration pages](#)

11. Once you click on the **Administration pages** link, you can then go to your **Status report** page through the **Reports | Status report** link.
12. The **Status report** should now show you the new version of Drupal 6.13 listed. It should also show you green checks next to the rest of your configuration. This shows you that you have successfully upgraded your core code to Drupal 6.13. Congratulations!

Status report

Operating in off-line mode.

Here you can find a short overview of your site's parameters as well as any problems detected with your installation. It may be useful to copy and paste this information into support requests filed on drupal.org's support forums and project issue queues.

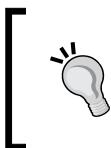
Drupal	6.13
 Access to update.php	Protected
 Configuration file	Protected
 Cron maintenance tasks	Last run 1 hour 19 min ago You can run cron manually .
 Database updates	Up to date
 File system	Writable (<i>public</i> download method)
 GD library	bundled (2.0.34 compatible)
 MySQL database	5.0.81
 PHP	5.2.9
 PHP memory limit	32M
 PHP register globals	Disabled
 Unicode library	PHP Mbstring Extension
 Update notifications	Not enabled Update notifications are not enabled. It is highly recommended that you enable the update status module from the module administration page in order to stay up-to-date on new releases. For more information please read the Update status handbook page .
 Web server	Apache/2.2.13 (Unix) mod_ssl/2.2.13 OpenSSL/0.9.8k DAV/2 mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2.2635

13. Run `cron` manually to make sure the Status report loads again without any errors. `cron` should run successfully.
14. We're now ready to complete the entire upgrade process by installing the new 6.x versions of our contributed modules, and our Zen theme and corresponding custom theme files.
15. We will then enable the update status module (that now comes packaged with Drupal 6.x) and that will complete the entire upgrade.

Upgrading contributed modules

We're now ready to update our contributed modules to their 6.x versions. Follow these steps:

1. Download all of the contributed module `tar.gz` files corresponding to the latest 6.x stable releases to a folder on your desktop. You may want to create a folder called `contrib._module_upgrades` to put the new releases in.
2. Once you have downloaded all the `tar.gz` files, extract them into your contributed modules folder. This will create all of your module folders (one for each module).



If you are using CCK and Imagefield, you'll also need to download the Filefield module – this is a dependency of Imagefield in Drupal 6.x.

3. Also, make sure you download and extract the Advanced Help module, so you can add advanced help documentation to your Drupal site (this is a large scale Drupal 6.x enhancement). The project page is here: http://drupal.org/project/advanced_help

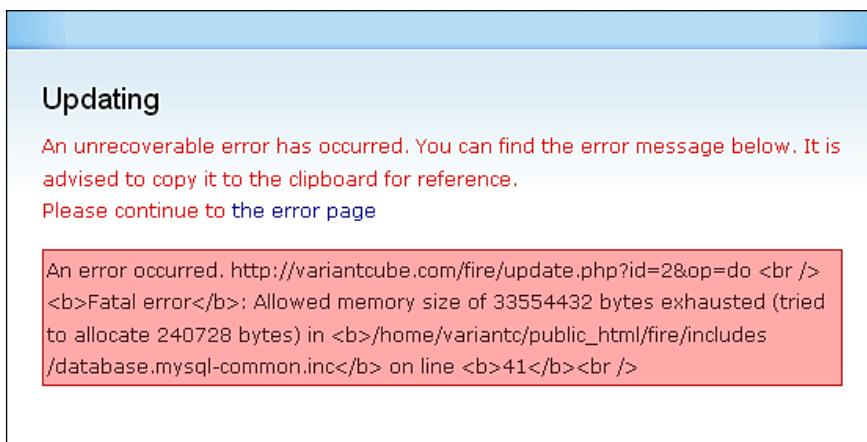
4. You should by now have extracted all of your modules ready to upload via FTP:

Name	Ext	Size	Type	Changed	Attr		Name	Ext
..			Parent directory	8/18/2009 11:46:23 AM			..	
views_bonus			File Folder	8/18/2009 11:45:58 AM				
views			File Folder	8/18/2009 11:45:59 AM				
panels			File Folder	8/18/2009 11:46:23 AM				
lightbox2			File Folder	8/18/2009 11:46:21 AM				
jquery_plugin			File Folder	8/18/2009 11:46:19 AM				
imce			File Folder	8/18/2009 11:46:17 AM				
imagefield			File Folder	8/18/2009 11:46:15 AM				
imagecache			File Folder	8/18/2009 11:46:13 AM				
imageapi			File Folder	8/18/2009 11:46:11 AM				
filefield			File Folder	8/18/2009 11:46:09 AM				
fckeditor			File Folder	8/18/2009 11:46:07 AM				
devel			File Folder	8/18/2009 11:46:06 AM				
ctools			File Folder	8/18/2009 11:46:04 AM				
cck			File Folder	8/18/2009 11:45:54 AM				
advanced_help			File Folder	8/18/2009 11:46:01 AM				
admin_menu			File Folder	8/18/2009 11:45:59 AM				
views-6.x-2.6.tar.gz		1,534,198	WinRAR archive	8/18/2009 11:44:10 AM	a			
views_bonus-6.x-1.0-beta4.tar.gz		15,011	WinRAR archive	8/18/2009 11:44:20 AM	a			
panels-6.x-3.0-rc1.tar.gz		355,424	WinRAR archive	8/18/2009 11:43:23 AM	a			
lightbox2-6.x-1.9.tar.gz		101,435	WinRAR archive	8/18/2009 11:43:11 AM	a			
jquery_plugin-6.x-1.10.tar.gz		32,011	WinRAR archive	8/18/2009 11:43:57 AM	a			
imce-6.x-1.2.tar.gz		69,523	WinRAR archive	8/18/2009 11:42:58 AM	a			
imagefield-6.x-3.1.tar.gz		27,258	WinRAR archive	8/18/2009 11:41:39 AM	a			
imagecache-6.x-2.0-beta9.tar.gz		63,999	WinRAR archive	8/18/2009 11:42:28 AM	a			
imageapi-6.x-1.6.tar.gz		23,769	WinRAR archive	8/18/2009 11:42:15 AM	a			
filefield-6.x-3.1.tar.gz		77,310	WinRAR archive	8/18/2009 11:41:49 AM	a			
fckeditor-6.x-1.4-rc1.tar.gz		182,883	WinRAR archive	8/18/2009 11:42:46 AM	a			
devel-6.x-1.16.tar.gz		163,998	WinRAR archive	8/18/2009 11:42:03 AM	a			
ctools-6.x-1.0-rc1.tar.gz		216,456	WinRAR archive	8/18/2009 11:43:30 AM	a			
cck-6.x-2.5.tar.gz		417,417	WinRAR archive	8/18/2009 11:41:21 AM	a			
advanced_help-6.x-1.2.tar.gz		79,326	WinRAR archive	8/18/2009 11:44:44 AM	a			
admin_menu-6.x-1.5.tar.gz		45,284	WinRAR archive	8/18/2009 11:44:34 AM	a			

5. Move your new contributed modules using FTP to your /sites/all/modules directory.
6. Go to your Modules admin list and re-enable the 6.x versions of the modules you just installed through their respective groups. For example, re-enable all of the Views modules and then run your update.php script. Then come back to the modules admin list and re-enable your Panels modules. Doing it in this way by enabling the modules as groups will help you to identify problems with your module upgrades and more easily diagnose problems than if you just re-enable all your contributed modules at the same time.
7. The only module you do not need to enable at this time is the Devel module. We're going to look at that module in detail in Chapter 3, so leave it disabled for now. Be sure to enable the Advanced Help module as well. Click on **Save configuration**.

8. You may receive a **Fatal Error** warning about your `memory_limit` size (in your PHP settings). If you receive this, go ahead to the next step and run your `update.php` script. We'll address our memory issue immediately after running `update.php`.
9. If you receive parse errors, immediately run `update.php` again. This time the script will update all of your contributed modules database schemas.
10. You will see your Drupal database update page again (at `update.php`). Click on the **Continue** button.
11. Click on **Update** on the next screen (the same as our `update.php` during core upgrade above).
12. When I run `update.php`, Drupal informs that there is not enough memory to run the upgrade. This is a performance issue due to the amount of modules we've added to our site and the resources these modules require. Before continuing our `update.php` we'll need to upgrade our `memory_limit`. We can do this by editing our site's `settings.php` file. These are screenshots of the two types of errors which may arise at this stage of the process:

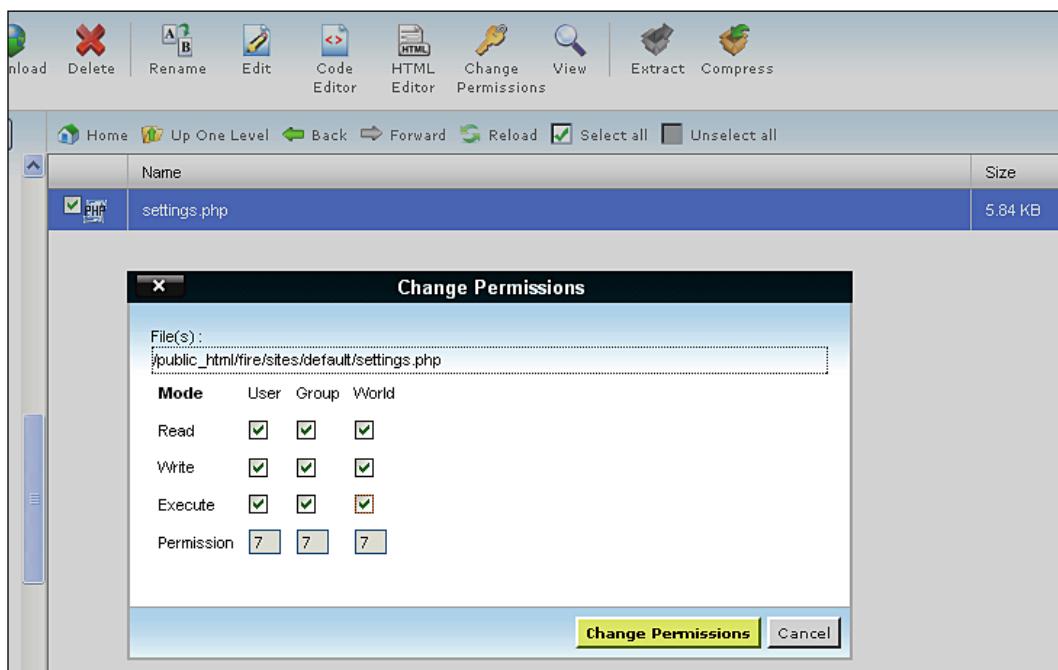
Fatal error: Allowed memory size of 33554432 bytes exhausted (tried to allocate 30720 bytes) in line 131



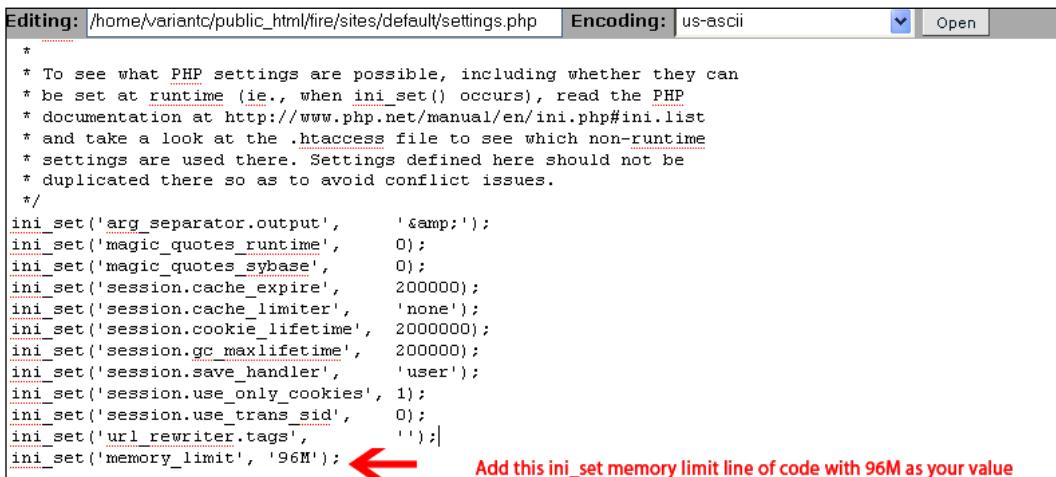
Updating your PHP memory limit

We're getting errors related to PHP memory, so we need to increase our `memory_limit` in our PHP settings. There are a number of methods of doing this. You can set a higher memory limit in your `.htaccess`, `php.ini` or `settings.php` files. We'll try doing this by adding a memory limit increase to our Drupal site's `settings.php` file first. There's more about tweaking Drupal PHP memory limits here: <http://drupal.org/node/207036>

1. Either login to your cPanel File Manager utility or to your site through FTP to edit the `settings.php` file that is in your `/sites/default` folder. If you do this via FTP, you will need to edit the file's permissions first in order to add or edit the file. It needs to have write permissions for editing purposes.
2. I've logged into my cPanel File Manager utility and I've browsed to my `/sites/default` directory.
3. I check the box next to my `settings.php` file and then click on the **Change Permissions** button in the File Manager admin screen. I add full write and execute permissions. I click on the **Change Permissions** button.



4. I then click on the **Edit** button in the File Manager utility.
5. This launches the `settings.php` file in edit mode. I look for the PHP settings section of the file and the array of `ini_set` settings. I add the following line of code to my file: `ini_set('memory_limit', '96M');`
6. I specify 96M as my new `memory_limit` value.



```
Editing: /home/variantc/public_html/fire/sites/default/settings.php Encoding: us-ascii Open
/*
 * To see what PHP settings are possible, including whether they can
 * be set at runtime (ie., when ini_set() occurs), read the PHP
 * documentation at http://www.php.net/manual/en/ini.php#ini.list
 * and take a look at the .htaccess file to see which non-runtime
 * settings are used there. Settings defined here should not be
 * duplicated there so as to avoid conflict issues.
 */
ini_set('arg_separator.output',      '&');
ini_set('magic_quotes_runtime',      0);
ini_set('magic_quotes_sybase',       0);
ini_set('session.cache_expire',     200000);
ini_set('session.cache_limiter',    'none');
ini_set('session.cookie_lifetime',  2000000);
ini_set('session.gc_maxlifetime',   200000);
ini_set('session.save_handler',     'user');
ini_set('session.use_only_cookies', 1);
ini_set('session.use_trans_sid',    0);
ini_set('url_rewriter.tags',        '');
ini_set('memory_limit', '96M');
```

7. Save the changes to your file.
8. Refresh your site admin screen to see if the changes worked.
9. If you refresh your `update.php` screen, the updates should automatically begin because your memory limit is now working. If you receive any remaining parse errors, run the `update.php` script again until you see no errors.
10. Drupal will give you the following note/info when you run `update.php` to upgrade your modules.



Please note that the Panels upgrade from Drupal 5 to Drupal 6 is far from perfect, especially where Views and CCK are involved. Please check all your panels carefully and compare them against the originals. You may need to do some rework to regain your original functionality.

This is why we took the time earlier to make notes about the home page panel we're using on the site, in case we need to rebuild it.

11. You should now see a final Drupal database update screen with links back to your Administration pages (just like at the end of the core upgrade process). Click on the **Administration pages** link.

Drupal database update

Updates were attempted. If you see no failures below, you may proceed happily to the [administration pages](#). Otherwise, you may need to update your database manually. All errors have been [logged](#).

- [○ Main page](#)
- [○ Administration pages](#)

12. If you have installed the CTools module (to allow Panels to work) you'll see two messages appear at the top of the Administer page: **The directory files/ctools has been created** and **The directory files/ctools/css has been created**.

- [○ Operating in off-line mode.](#)
- [○ The directory files/ctools has been created.](#)
- [○ The directory files/ctools/css has been created.](#)

13. Load your **Status report** page and you should see everything in green mode with checks. You've successfully completed stage 1 of the contributed module upgrade process. Congratulations! Notice that on your **Status report** page you'll see the PHP memory limit is now set to **96M**.

Drupal	6.13
✓ Access to update.php	Protected
✓ CTools CSS Cache	Exists
✓ Configuration file	Protected
✓ Cron maintenance tasks	Last run 1 sec ago You can run cron manually .
✓ Database updates	Up to date
✓ File system	Writable (<i>public</i> download method)
✓ GD library	bundled (2.0.34 compatible)
✓ GD library	bundled (2.0.34 compatible)
✓ MySQL database	5.0.81
✓ PHP	5.2.9
✓ PHP memory limit	96M
✓ PHP register globals	Disabled
✓ Unicode library	PHP Mbstring Extension

14. Run cron manually once more to make sure you do not receive any errors. The last items to address will be to re-enable your View for the photo gallery and your home page panel. Finally, we'll re-install and enable your custom Zen sub-theme.

Installing the updated Zen theme files

We're going to install and upgrade the contributed Zen theme (6.x-1.0) so that we can reinstall and configure our Apollo custom theme files. The first part of this process is to install the main Zen theme folder in your /sites/all/themes directory.

1. Download the 6.x version of the Zen theme from its project page here: <http://drupal.org/project/zen>
2. Extract the file in your contributed modules directory on your desktop.
3. Connect to your site through FTP and browse to the /sites/all/themes folder.
4. Upload the Zen folder.
5. Now when you refresh your themes admin list, you'll see the 6.x-1.0 versions of Zen, Zen Classic, and the Zen Themer's Starter Kit.

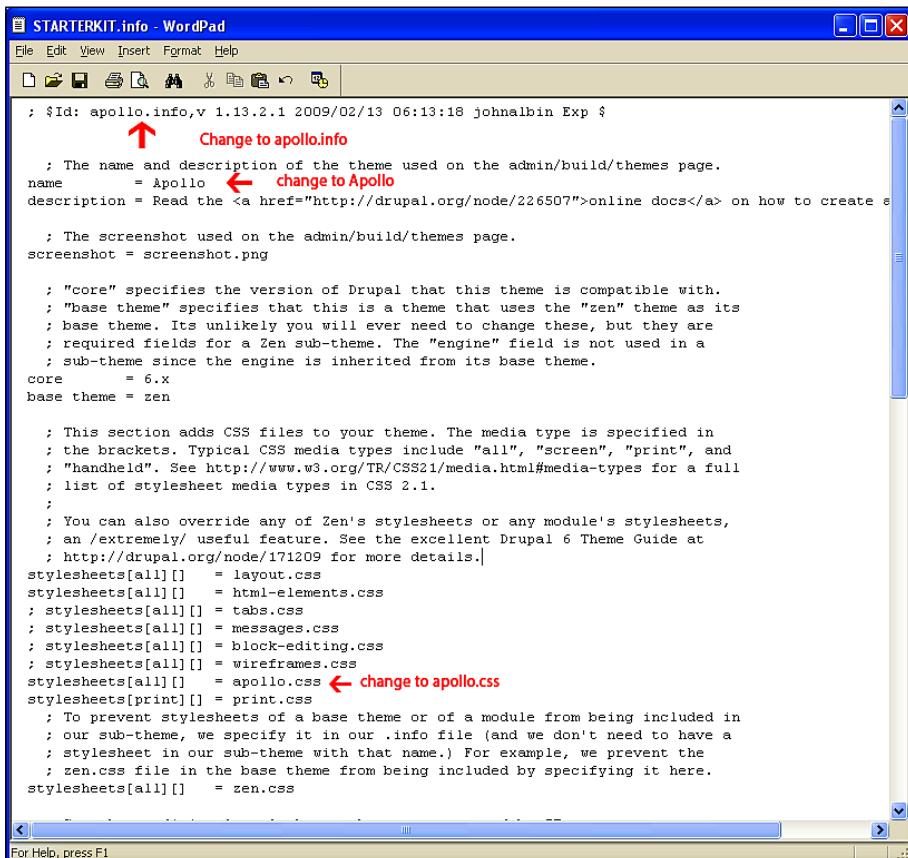
The screenshot shows the Drupal 6.x themes administration page. It lists three themes:

- Zen**: Described as "the ultimate starting theme for Drupal 6". Version 6.x-1.0. Status: Enabled (checkbox checked). Action: configure.
- Zen Classic**: Described as "Zen sub-theme based on Deliciously Blue.". Version 6.x-1.0. Status: Disabled (checkbox unchecked). Action: none.
- Zen Themer's Starter Kit**: Described as "Read the online docs on how to create a sub-theme.". Version 6.x-1.0. Status: Enabled (checkbox checked). Action: none.

Upgrading your custom theme

Now we need to add our custom theme back to the site. Our theme is called 'Apollo' and we originally used the Zen Themer's Starter Kit files in Drupal 5.x to create this custom theme. Here are the steps for upgrading our custom theme to Drupal 6.x and for using the new upgraded Zen Themer's Starter Kit code base. Bear in mind that we did not make any template code edits to our original Apollo theme, so we do not need to worry about updating our new template files with our existing template code. This may be different in your specific case. Make sure to have good backups of all your existing Drupal 5.x custom theme files. We will need to copy over the CSS files from our existing Apollo theme, so we have the same CSS styles in our new 6.x site. Here are the steps:

1. Copy over your existing `apollo` folder using FTP into the main `Zen` directory in your `/sites/all/modules` folder.
2. Open the `apollo` folder on the remote side and delete the following files: `template.php`, `theme-settings.php`, and `theme-settings.init.php`. These are the older template files and we will replace these with the latest TPL versions – make sure you note down any specific theme override functions you may have customized or added – you can copy those out and then paste them back into the new TPL files once you add them. In our case, we did not add any specific theme overrides.
3. Now, with Drupal 6.x themes the new rule is that you need to have a `.info` file in your theme directory, so that the theme will show up in the modules admin list, and in order for all of your theme regions to be defined properly. You need to add an `apollo.info` file in the root directory of the theme. Copy the `STARTERKIT.info` file (from the `STARTERKIT` theme folder) to your local side of FTP. Open the file in WordPad, and change any reference to `STARTERKIT` in the file to `apollo`.



```

; $Id: apollo.info,v 1.13.2.1 2009/02/13 06:13:18 johnalbin Exp $
↑          Change to apollo.info

; The name and description of the theme used on the admin/build/themes page.
name      = Apollo ← change to Apollo
description = Read the <a href="http://drupal.org/node/226507">online docs</a> on how to create a theme.

; The screenshot used on the admin/build/themes page.
screenshot = screenshot.png

; "core" specifies the version of Drupal that this theme is compatible with.
; "base theme" specifies that this is a theme that uses the "zen" theme as its base theme. Its unlikely you will ever need to change these, but they are required fields for a Zen sub-theme. The "engine" field is not used in a sub-theme since the engine is inherited from its base theme.
core      = 6.x
base theme = zen

; This section adds CSS files to your theme. The media type is specified in the brackets. Typical CSS media types include "all", "screen", "print", and "handheld". See http://www.w3.org/TR/CSS21/media.html#media-types for a full list of stylesheet media types in CSS 2.1.
;

; You can also override any of Zen's stylesheets or any module's stylesheets, an /extremely/ useful feature. See the excellent Drupal 6 Theme Guide at http://drupal.org/node/171209 for more details.
stylesheets[all][] = layout.css
stylesheets[all][] = html-elements.css
; stylesheets[all][] = tabs.css
; stylesheets[all][] = messages.css
; stylesheets[all][] = block-editing.css
; stylesheets[all][] = wireframes.css
stylesheets[all][] = apollo.css ← change to apollo.css
stylesheets[print][] = print.css
; To prevent stylesheets of a base theme or of a module from being included in our sub-theme, we specify it in our .info file (and we don't need to have a stylesheet in our sub-theme with that name.) For example, we prevent the zen.css file in the base theme from being included by specifying it here.
stylesheets[all][] = zen.css

```

4. Save the file and then re-name it to `apollo.info`. Upload this `apollo.info` file back to your `apollo` theme directory.
5. Refresh your themes admin list and the Apollo theme will now be visible.
6. Grab the `template.php` file and the `theme-settings.php` files from the `STARTERKIT` folder and move those to your local site. Rename any instance of `STARTERKIT` in those 2 files to `apollo`. This includes the main function in `template.php` (function `apollo_theme`):

```
/**  
 * Implementation of HOOK_theme().  
 */  
function apollo_theme(&$existing, $type, $theme, $path) {  
  $hooks = zen_theme($existing, $type, $theme, $path);  
  // Add your theme hooks like this:  
  /*  
   $hooks['hook_name_here'] = array( // Details go here );  
  */  
  // @TODO: Needs detailed comments. Patches welcome!  
  return $hooks;  
}
```

This is the renamed function in `theme-settings.php`:

```
function apollo_settings($saved_settings) {  
  // Get the default values from the .info file.  
  $defaults = zen_theme_get_default_settings('apollo');  
  // Merge the saved variables and their default values.  
  $settings = array_merge($defaults, $saved_settings);  
  /*  
   * Create the form using Forms API: http://api.drupal.org/api/6  
  */
```

7. Once those functions are renamed, upload those two PHP files to your `/apollo` directory.

8. Refresh your themes admin listing.
9. Enable your **Apollo** theme as the **Default**.

Screenshot	Name	Version	Enabled	Default	Operations
	Apollo Read the online docs on how to create a sub-theme.	6.x-1.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

10. Confirm that your Garland theme is set to the default admin theme here: **Administer | Site Configuration | Administration theme**. You can also choose whether you want to use the Garland theme for content editing. Save configuration.
11. Congratulations! You've upgraded your custom theme to Drupal 6.x. Good work!

Cleaning up and resetting Views

The last thing to do in order to have a completely successful upgrade process is to carry out any content cleanup that's required and to reset your Views. Looking at our photo gallery content pages post-upgrade, I notice that the Lightbox2 imagecache settings were not implemented during the upgrade process. In order to use the correct **Teaser** and **Full node** display types, you may need to go back into your Photo Gallery content type and manage the **Display fields** configuration on your Photo file field.

The last thing you need to do is import your View code that we had exported from our Drupal 5.x site before we upgraded. Go ahead and open that .txt file you saved and copy all the code.

1. Select all and copy code from your `photo_gallery.txt` file.
2. Go to your **Site Building | Views** admin and click on the **Import** button.
3. Leave the **View name** field blank.
4. Paste your copied View export code into the box.

5. Click on **Import**.

The screenshot shows the Drupal Views2 interface. At the top, there is a breadcrumb navigation: Home > Administer > Site building > Views. Below the breadcrumb, there is a horizontal menu with tabs: Views, List, Add, Import (which is highlighted in blue), and Tools. The main content area has two sections: "View name (key=name, weight=0):" with an input field containing a placeholder, and "Paste view code here (key=view, weight=0): *". The code area contains a large block of PHP code for defining a view named 'photo_gallery'. At the bottom of the code area is a scroll bar. Below the code area is a button labeled "Import".

```
$view = new stdClass();
$view->name = 'photo_gallery';
$view->description = 'Photo Gallery';
$view->access = array (
);
$view->view_args_php = '';
$view->page = TRUE;
$view->page_title = 'Antique Fire Apparatus & Muster, Harrisburg, PA';
$view->page_header = '';
$view->page_header_format = '1';
$view->page_footer = '';
$view->page_footer_format = '1';
$view->page_empty = '';
$view->page_empty_format = '1';
$view->page_type = 'bonus_grid';
$view->url = 'photo_gallery';
$view->use_pager = TRUE;
$view->nodes_per_page = '24';
$view->sort = array (
  array (
    'tablename' => 'node',
    'field' => 'created',
    'operator' => 'desc',
    'weight' => '0',
  ),
);
```

6. Check your View settings in the new Views2 interface to make sure they're translated correctly.
7. Save your View.
8. Test your View as a page and a block.

Placing your site back online

Now that you have completed the upgrade process, you can place your site back online. Browse to **Administer | Site configuration | Site maintenance** and check the **Online** radio button. Save configuration and your site will return to online status.

Home > Administer > Site configuration

Site maintenance

Site status (key=site_offline, weight=0):

Online
 Off-line

When set to "Online", all visitors will be able to browse your site normally. When set to "Off-line", only users with the "admin:configuration" permission will be able to access your site to perform maintenance; all other visitors will see the site off-line message configured below. Authorized users can log in during "Off-line" mode directly via the [user login](#) page.

Site off-line message (key=site_offline_message, weight=0):

Fire Truck Photos is currently under maintenance. We should be back shortly. Thank you for your patience.

Message to show visitors when the site is in off-line mode.

[Save configuration](#) [Reset to defaults](#)

Summary

Congratulations! You have successfully upgraded your Drupal site. Here's a brief recap of what you did in this chapter:

- After backing up your site and database, taking your site offline and running Status report, you upgraded your Drupal core code to Drupal 5.19.
- You installed the Drupal Update Status module and later removed the module so that it would not cause conflicts during the 6.x upgrade.
- You upgraded all of your 5.x versions of contributed modules – a very important step prior to the upgrade to Drupal 6.x.
- You backed up your Views by exporting their code.
- You prepped for the Drupal 6.x upgrade by disabling all contributed modules and enabling the default Drupal Garland theme.
- You upgraded the Drupal core code to 6.13 and ran update.php to update the entire Drupal database schema, thereby confirming that your MySQL database tables were updated.
- You made your first major performance tweak to your server and site by raising the PHP `memory_limit` settings to 96M – this setting needs to be larger to run Drupal 6.x and all of the contributed modules you have.

At this point you can bring your site back online using the site maintenance configuration. We'll see you back here for more performance tips in Chapter 2, including how to manage the Drupal cache system, clear the theme registry, and tweak your PHP settings.

2

Maintaining your Drupal Site

Now that you have upgraded to Drupal 6.13, you're ready to learn how to maintain the site, keep it running smoothly on a regular basis, and enable some performance enhancements that we will continue to monitor. This chapter will show you a selection of best practice Drupal maintenance tips and tricks that you can enable using the core Drupal administration interface. These performance enhancements will help you to run a powerful and well-maintained Drupal site.

Running a Drupal website is like keeping a fire truck prepared to go on a rescue call. The apparatus needs to be washed and inspected daily by firefighters so that it's ready to go fight a fire or rescue someone in need at a moment's notice. Your Drupal site also needs to be closely taken care of daily so that it runs smoothly, leaving less room for issues both on the site frontend and the backend, including the server.

By the end of this chapter, you will be able to perform the following maintenance:

- Inspect your Drupal configuration file.
- Check PHP and MySQL configuration using the Status report.
- Delete files you no longer need in your Drupal directory post upgrade.
- Enable the Drupal 6.x core Drupal Update Status module.
- Disable and uninstall contributed modules that are inactive.
- Enable and configure the Drupal performance cache.
- Clear your Drupal performance cache and clear the theme registry so that you can view theme tweaks immediately.
- Run cron tasks manually by using the Status report.
- Run cron tasks automatically using the Poormanscron module.
- Set up server side cron tasks using the cPanel admin.

- Tweak your PHP.ini settings including the memory_limit, upload_max_size, and post_max_size, and check which PHP extensions are enabled or disabled.
- Tweak your HTACCESS settings and configuration.
- Back your site up using phpMyAdmin and SFTP via cPanel. Through direct remote access, keep regular site directory and database backups, and use contributed backup modules.

By the conclusion of the chapter, you will develop a new skill set as a Drupal maintenance technician, ready to maintain your own Drupal site on an hourly and daily basis. You've already learned how to upgrade your site, so let's learn how to keep a close watch on the site and maintain its ongoing growth and development.

Checking your Drupal configuration status

Now that you have successfully upgraded to Drupal 6.13, you can run your **Status report** to check on the status of your configuration file and other site components such as the PHP and MySQL versions; GD library version, memory_limit, and whether or not your modules are updated. **Status report** will give you the most up-to-date checklist on the general performance of your site. When we run **Status report** we find out the following:

- Our update.php file is protected. The permissions on this file are 644, meaning that read permissions are granted to user, group, and world, while write and execute permissions are enabled only for the users of the file.
- CTool CSS Cache is in use and exists. This is the CSS cache for the CTools module that is used in conjunction with Panels.
- Our main site settings.php file is protected. This has the same permissions level as the update.php file: read-only permissions.

It's important to emphasize here that your settings.php configuration file should always be set to read-only when your site is online and publicly accessible. If you download the file and view it in a text editor, you'll see this information at the top of the file under the **IMPORTANT NOTE** section. The file also tells you where it's located in your Drupal directory. For example, if you are running one Drupal site then the file is located in the /sites/default folder. If you are running a multisite installation, the file may be located in the /sites/nameofsite.org/ folder. With multisite installations, each site will have its own settings.php file in its root directory.

The next set of information in the configuration file specifies your database settings. With Drupal you can integrate your site with one database (the common install method) using one \$db_url line of code. You can also specify multiple databases if you need your Drupal site to connect to multiple databases (for example, if you're running CiviCRM you could set the CiviCRM database as the second database to use). Finally, you can specify database table prefixes. This is a possible configuration if you need your Drupal site to have a contributed module like Views2 Integrate with another module such as CiviCRM – there are times when this type of db_prefix is necessary.



See the following articles on <http://drupal.org/> and CiviCRM for more information. Views2 Integration Module: <http://wiki.civicrm.org/confluence/display/CRMDOC/Views2+Integration+Module>. How to configure multiple databases in your settings.php file: <http://drupal.org/node/18429>

As this article states, the code you'll use depends on whether you're connecting to one database or multiple. The default Drupal installation has one line of code:

```
<?php  
$db_url = 'mysql://dbusername:dbuserpassword@localhost/dbname';  
?>
```

If you are connecting to multiple databases, you'll need to add another db_url line of code. Set your main Drupal database as default. For example:

```
<?php  
$db_url['default'] =  
  'mysql://dbusername:dbuserpassword@localhost/dbname';  
$db_url['db2'] = 'mysql://dbuser:pwd@localhost/anotherdbname';  
$db_url['db3'] = 'mysql://dbuser:pwd@localhost/yetanotherdbname';  
?>
```

You may need to configure this type of setup if you are trying to integrate a module such as CiviCRM with a Drupal module such as Views2. You need to set up the Drupal database user to have SELECT access to the CiviCRM database as well, and then add a specific array to your \$db_prefix. For example, this array may look something like the one in this article: <http://wiki.civicrm.org/confluence/display/CRMDOC/Views2+Integration+Module>.

The next section in your configuration file is your Base URL – this should be specified though it's not required. In this section of the file, we'll uncomment the `$base_url` line of code and add our website's URL. In this case it will be:

```
$base_url = 'http://variantcube.com/fire'; // NO trailing slash!
```

Setting the `base_url` becomes more important when installing Drupal in a multisite configuration where you have multiple Drupal instances running off one core Drupal install. In this case you need to set the `base_url` for each of your multisites so that each site runs correctly when a user types in that site's subdomain URL. In our case the `base_url` is not required and you can simply leave this line of code commented out. You can also uncomment it as per my instructions above.

To make changes to your `settings.php` file you'll need to change the permissions on the file to write permissions. Use SFTP or your cPanel file manager utility. Then edit the file and make your changes. Save the file, re-upload it to your server, and then immediately refresh your site to make sure everything is still loading correctly. Then reset the permissions to read-only. When editing the permissions on the file using an FTP file manager utility, you set them to 777 or 755. Once you complete the edits you can restore the permissions to 555.

The next section of your configuration file contains PHP settings. We'll look at this in the next tip. Your **Status report** also provides the following information:

- When your last cron run was engaged (in minutes).
- Whether your database schema is up-to-date.
- If your filesystem is writable by the Drupal system.
- What version of the GD imaging library is loaded and whether you have enough site memory to load it.
- The versions of MySQL and PHP installed on your server.
- Your PHP `memory_limit`. You will recall that in Chapter 1 we increased this memory limit to 96M so that we could successfully install our new upgrade and run all of our contributed modules. 96M tends to be the recommended default memory limit for a Drupal 6.x site. When you install contributed modules, such as Views, Panels, or CiviCRM, you may need to raise your PHP `memory_limit` to 96M, or the modules will not work or perform well.
- A specification that `register_globals` is disabled. In order to run Drupal, your host/server will be required to have this PHP setting disabled. If it's enabled, you will receive an error when you try and install or upgrade Drupal.
- The version of your web server (we're running Apache 2.2.13 here).

Checking your PHP and MySQL settings

You can also access your PHP and MySQL configuration settings via the Drupal **Status report**. For PHP it's as simple as clicking on the PHP version number, which is hyperlinked in your **Status report**. The same goes for your MySQL version and settings.

✓ MySQL database	5.0.81
✓ PHP	5.2.9

Clicking on the PHP version link loads a `php.info` file that resides in your site. This will give you all of your PHP core configuration settings and all of the PHP extensions you have loaded and enabled on your server. It's good to review this file for the following information that you'll need as you troubleshoot performance.

The location of your loaded `php.ini` configuration file: It's good to know where the default `php.ini` file is located on your web server. You can overwrite this configuration with a custom `php.ini` file or with custom PHP setting code that you load into either your `settings.php` and/or `.htaccess` file. However, there may be times when you need to edit the original default `php.ini` file as long as your host server gives you write access to that file.

Specifications under the **PHP Core configuration** section of the information file note the following information:

- Whether `file_uploads` is enabled/on. You will need to confirm that `file_uploads` is On in your PHP settings in order to use Drupal's core file attachments module.
- `Max_execution_time` and `max_input_time`.
- `Memory_limit`: notice its telling us that the local value of the `memory_limit` is 96M, which is what we set by adding a line of custom code to our `settings.php` file in Chapter 1. The Master value for `memory_limit` is still set server wide to 32M, but our site is using 96M. The Master value is being forced by the default server `php.ini` file. When we added our line of code to our local site's `settings.php` file, it only made the new PHP `memory_limit` effective for our site. The local value will be used in your site and will override any master value set.

- `Post_max_size` and `upload_max_filesize`. These settings may need to be tweaked once our users start uploading files to our website. Drupal allows you to control the file sizes per upload (per file) and per user through the Drupal administration configuration. When you make a change to the maximum upload file size (per file and per user) in your Drupal configuration, you need to make sure that these tweaks have also been made in your PHP settings first. So, if you want to raise your maximum upload size per file to 30 MB, you need to make sure to increase your PHP setting for `upload_max_filesize` to 30 MB respectively.

The important thing to grasp now is that you have access to review these PHP settings through your site's Status report and this PHP information file becomes an invaluable tool to troubleshoot the performance of your Drupal site. For example, if a user of your site complains that they are receiving errors every time they try and post a 50 MB PowerPoint file or TIFF image, you can check your PHP settings to see what the `upload_max_filesize` is, and then make a local tweak by adding a line of code to your `settings.php` file or by adding a custom `php.ini` file to your server. We'll try this as an example later in this chapter.

<code>memory_limit</code>	96M	← Local value	32M
<code>open_basedir</code>	<i>no value</i>	<i>no value</i>	
<code>output_buffering</code>	<i>no value</i>	<i>no value</i>	
<code>output_handler</code>	<i>no value</i>	<i>no value</i>	
<code>post_max_size</code>	8M		8M
<code>precision</code>	12		12
<code>realpath_cache_size</code>	16K		16K
<code>realpath_cache_ttl</code>	120		120
<code>register_argc_argv</code>	On		On
<code>register_globals</code>	Off		Off
<code>register_long_arrays</code>	On		On
<code>report_memleaks</code>	On		On
<code>report_zend_debug</code>	On		On
<code>safe_mode</code>	Off		Off
<code>safe_mode_exec_dir</code>	<i>no value</i>	<i>no value</i>	
<code>safe_mode_gid</code>	Off		Off
<code>safe_mode_include_dir</code>	<i>no value</i>	<i>no value</i>	
<code>sendmail_from</code>	<i>no value</i>	<i>no value</i>	
<code>sendmail_path</code>	/usr/sbin/sendmail -t -i		/usr/sbin/sendmail -t -i
<code>serialize_precision</code>	100		100
<code>short_open_tag</code>	On		On
<code>SMTP</code>	localhost		localhost
<code>smtp_port</code>	25		25
<code>sql.safe_mode</code>	Off		Off
<code>track_errors</code>	Off		Off
<code>unserialize_callback_func</code>	<i>no value</i>	<i>no value</i>	
<code>upload_max_filesize</code>	2M	← Local value	2M
<code>upload_tmp_dir</code>	<i>no value</i>	<i>no value</i>	
<code>user_dir</code>	<i>no value</i>	<i>no value</i>	
<code>variables_order</code>	EGPCS		EGPCS
<code>xmlrpc_error_number</code>	0		0
<code>xmlrpc_errors</code>	Off		Off
<code>y2k_compliance</code>	On		On
<code>zend.ze1_compatibility_mode</code>	Off		Off

Files to delete and clean up

Your Status report tells you that your `update.php` file is protected. This is good as you do not want anyone to launch your Drupal site and try running `update.php` to update your database schema. You only want to do this as a super user admin. Another tip is to delete your `install.php` file from your Drupal site. You do not need to keep the `install.php` file on your production site. Removing it is a good precaution and the added benefit is that no unauthorized site user can try and run your `install.php` file. To do this:

1. SFTP into your site or access through cPanel.
2. Copy a backup of your `install.php` to your local backup folder so that you have it if you need it later.
3. Select your `install.php` file and delete it.

This would also be a good time to review your site's `/sites/modules` and `/sites/themes` folders, and make sure you do not have any residual files or `.tar.gz` files remaining post upgrade. If you do, you can safely remove them from your site. You do not want to have module folders in your site if you don't need them because they will just take up space, which in turn makes your site/server sluggish.

Enabling the Update Status module

Remember that when we ran our Drupal 5.x site, we had to install the contributed Update Status module in order to get status notifications on our site's modules. This module now comes packaged with Drupal 6.13 core. So all we need to do is enable it. We see the module is currently disabled by checking our Status report and viewing the notification that the module is disabled:

 **Update notifications** Not enabled

Update notifications are not enabled. It is **highly recommended** that you enable the update status module from the [module administration page](#) in order to stay up-to-date on new releases. For more information please read the [Update status handbook page](#).

1. Visit your modules admin page here: <http://variantcube.com/fire/admin/build/modules>
2. Under your Core-optional module list, look for **Update status** and check the box next to the module to enable it. Save configuration.

Revisit your Status report and you should receive new notifications of modules or themes that need updating. Update Status will flag two types of updates in its notification system. The first type of update will be shown marked with a yellow background in your Available updates table. Yellow flagged updates are recommended version upgrades. These are usually new versions of the module that include potential new module functionality. They are not required as they do not include security patches to the module.

The second type of update is marked in dark red with a pastel red background color. These are required security patch upgrades that you should run as soon as possible after receiving the notification.

Notice that we have a new security update notification on our site displayed with the red background and red text. The Drupal Update Status module has located a new version of **ImageCache (6.x-2.0-beta9)**. This is a security patch release so it's a required patch we'll need to download in the near future.

A screenshot of a web page showing a security update notification for the ImageCache module. The notification has a red background and contains the following text:

ImageCache 6.x-2.0-beta9

Security update: 6.x-2.0-beta10 (2009-Aug-19)

Includes: *ImageCache, ImageCache UI*

Security update required! X

Download Release notes

Download the latest 6.x version of Imagecache and upload the new module folder to your server to replace the existing outdated module. You may need to run update.php again if the module makes any updates on your MySQL database tables. The latest Imagecache module is here: <http://drupal.org/project/imagecache>.

Disabling unused modules and themes

Now that you have your Drupal 6.13 site up and running, it's a good idea to review the core and contributed modules you're using, and to disable those modules and themes that are not being actively used. Disabling the modules will let Drupal know that it does not need to load those module hooks when they are not being used on the site. Disabling the module will keep the module in your site for future use and will keep that module's data in your database, but it will not be loaded during page loads or any other Drupal functionality since it's been disabled.

You can also choose to uninstall a module completely from your site if you are not using it, but bear in mind that if you uninstall a module in Drupal, it will remove not only the module but all the data associated with it. So, be sure to make a full backup of your site and database before removing any modules for good.

You can also disable all core modules and themes that you are not using. Check which these are, disable them, and this will help to speed up your website.

1. Go to your modules admin list at: <http://variantcube.com/fire/admin/build/modules>.
2. Uncheck any modules not being used. If you uncheck a contributed module you're no longer using, and do not need the module at all in your site, you can proceed to uninstall it.
3. For example, currently in my site I'm not using the sub-module of CCK called **Node Reference** nor the **User Reference** module. So I'm going to disable these. I may use them in the future, so I will leave them on my site in case I decide to re-enable them later.
4. I'm also not going to allow my users to change the colors of the themes, so I'm going to disable the core **Color** module. Go ahead and disable any specific modules you are not using.

You can also disable any core or contributed themes you are not planning to use following these instructions. Load the themes admin page here:

<http://variantcube.com/fire/admin/build/themes> and make sure you only have the public and admin themes you're using enabled. In our case on the /fire site, we're using **Garland** as the admin theme so that's enabled and we're using **Apollo** as the **Zen** sub-theme. I will also leave the main **Zen** theme enabled. I can safely disable the other themes.

The good thing about doing this type of maintenance is that it prevents your users (both admin and public) from being able to access those additional themes.

Introduction to Drupal caching

Drupal gives you various methods of caching your site's data and content by using the Drupal core administrative interface. There are a variety of contributed modules that allow for more advanced caching (we'll look at these in later chapters). Drupal allows you to cache data and content in order to speed up the performance of your site in terms of how quickly your pages and entire site loads for the end user.

Caching as much data and content as possible, especially the content that you show to your anonymous site visitors which includes content, blocks, and menus that may not change frequently, will help Drupal to speed up page load times on your site. Drupal will keep the cached data stored in a temp location either on the server or in the MySQL database. The site can easily fetch it for load time from that location.

Drupal does this by storing cached data in specific database tables of your MySQL database, so it can easily retrieve the cached data instead of recomputing, reloading, and reprocessing menu, block, and theme data each time the page loads. For example, your Drupal site may have 50 nodes of content. Each time someone visits your site they may load 5 of those nodes. The next time a visitor comes to your site and loads the same 5 nodes, Drupal has to process those nodes a second time, and each successive time for each visitor. So, what Drupal allows you to do is cache this data so that the node only loads once.

Drupal stores the cached version of the page in the cache table of the database. This way the next time a visitor launches that page, menu, or block item, they are loading the cached page. This ultimately speeds up the entire load process and load times of your site.

Drupal caches the following content and data:

- Sessions
- Variables
- Menus and blocks
- Nodes (pages)
- Views

There are times, however, when you do not want to cache data and content on your site. This is what we're going to focus on now in this chapter. We want to take a look at the methods Drupal gives us for controlling our performance caching on our site, and the different options we can select for caching when we're, both, in development of our site and in production 'live' site environments. Other issues can come up with cached pages. If you are actively developing a site and working on content, when you make a change to a node and the site visitor loads a cached version of that node, they will not see your active changes to that node. So, this could be a drawback. We'll explore these different options and see how we can easily clear our performance cache, and also set the cache for a specific minimum lifetime so that you can expire a performance cache on a specific set of Drupal content.

Enabling and configuring Drupal caching

We will first explore the Drupal cache options by looking at what Drupal core gives us out of the box in terms of caching and performance options to control and configure.

To access your core performance cache administrative page go to **Site configuration | Performance**. This will load the main Performance admin page.

The first section of this page is devoted to Page cache. These settings control how your Drupal node content is being cached and stored by the Drupal system. There are 3 caching modes and you must choose one option as the admin or developer of the site. By default the option for **Disabled** caching will be set:

- Disabled
- Normal
- Aggressive

The default has caching set to **Disabled** on the site. The recommended setting is to have normal caching mode enabled. Normal mode is fine to use for production level websites. As Drupal explains, normal mode does not cause side effects on your site or with your modules.

Aggressive mode caching can potentially cause side effects. It's a trade off, as aggressive caching causes a significant performance boost on your site, but it can also interfere with your content loading correctly and your contributed modules loading correctly. When you enable some contributed modules, they may not be compatible with aggressive caching mode, so be sure to check this page every time you enable a new module.

We're going to enable the **Normal** caching mode by clicking on the radio button to select it. We are also going to set our **Minimum cache lifetime** to 3 hours. This means that every 3 hours our cache will be cleared by Drupal, but if content is edited on the site within 3 hours from when the cache was last cleared, your users may not see that new content, menu item, or block item. We will also enable **Page compression**, which will cause the pages to load faster and save bandwidth on our web server. You may want to check with your hosting company to see if they already perform this type of optimization on your site/server before enabling it through your Drupal admin.

Performance

Page cache

Enabling the page cache will offer a significant performance boost. Drupal can store and send compressed cached pages requested by *anonymous* users. By caching a web page, Drupal does not have to construct the page each time it is viewed.

Caching mode:

Disabled
 Normal (recommended for production sites, no side effects)
 Aggressive (experts only, possible side effects)

The normal cache mode is suitable for most sites and does not cause any side effects. The aggressive cache mode causes Drupal to skip the loading (boot) and unloading (exit) of enabled modules when serving a cached page. This results in an additional performance boost but can cause unwanted side effects.

Currently, all enabled modules are compatible with the aggressive caching policy. Please note, if you use aggressive caching and enable new modules, you will need to check this page again to ensure compatibility.

Minimum cache lifetime:

On high-traffic sites, it may be necessary to enforce a minimum cache lifetime. The minimum cache lifetime is the minimum amount of time that will elapse before the cache is emptied and recreated, and is applied to both page and block caches. A larger minimum cache lifetime offers better performance, but users will not see new content for a longer period of time.

Page compression:

Disabled
 Enabled

By default, Drupal compresses the pages it caches in order to save bandwidth and improve download times. This option should be disabled when using a webserver that performs compression.

We will also enable our **Block cache** so that our block content and blocks load from a cached version instead of reloading them each time a user loads a Drupal site node. Notice that Drupal tells us here that if we also have page cache enabled, the performance of the block cache will mostly affect authenticated users on the site. Let's go ahead and enable **Block cache** mode. Also, note that if we have block content access restrictions in place, for example, if we have a block only showing to a specific role, the caching will be disabled for that block at that point in the process.

Block cache

Enabling the block cache can offer a performance increase for all users by preventing blocks from being reconstructed on each page load. If the page cache is also enabled, performance increases from enabling the block cache will mainly benefit authenticated users.

Block cache:

Disabled
 Enabled (recommended)

Note that block caching is inactive when modules defining content access restrictions are enabled.

If you have a theme that contains a large amount of CSS files being called and loaded from different folders of your theme folder, you may want to consider enabling optimization of CSS files under **Optimize CSS files**. Drupal will load all of the CSS code in one temporary cached CSS file as opposed to loading each individual CSS file separately. This can also improve overall site performance and load times. Beware, however, that caching CSS will cause issues while you're in development of the site, and you may not see your CSS changes/tweaks if you have this enabled while working on your theme. It's best to disable this before working on your theme files. Because our **Zen** sub theme is using about 4 different stylesheets, let's go ahead and enable this optimization and see how it affects our site.

Finally, let's enable our **JavaScript file optimization** as well. Again, beware that this can also cause interference with our contributed modules while in development, so only enable this when you are taking your site to a production environment.

Bandwidth optimizations

Drupal can automatically optimize external resources like CSS and JavaScript, which can reduce both the size and number of requests made to your website. CSS files can be aggregated and compressed into a single file, while JavaScript files are aggregated (but not compressed). These optional optimizations may reduce server load, bandwidth requirements, and page loading times.

These options are disabled if you have not set up your files directory, or if your download method is set to private.

Optimize CSS files:

Disabled
 Enabled

This option can interfere with theme development and should only be enabled in a production environment.

Optimize JavaScript files:

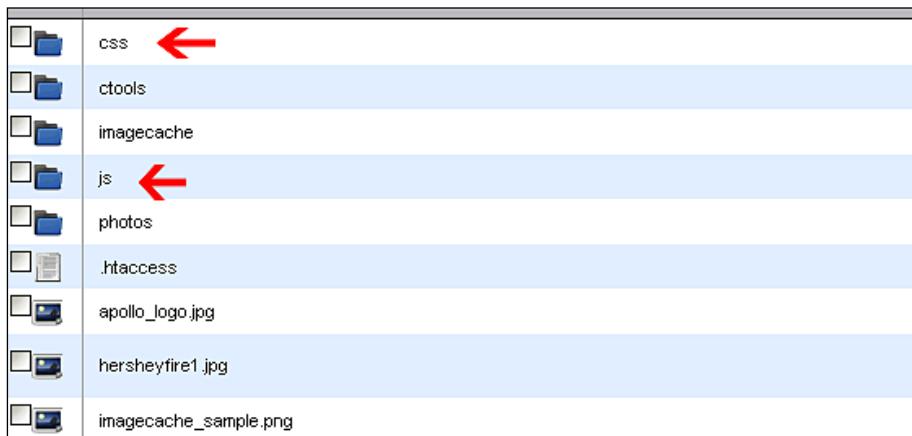
Disabled
 Enabled

This option can interfere with module development and should only be enabled in a production environment.

Save your configuration.

Once enabled, your caching and optimization settings will take effect and Drupal will be in caching mode until the minimum cache lifetime expires or until you disable the caching manually.

Additionally, 2 folders will be created in your /files folder. One is for CSS and one for your JavaScript cached files. Your cached temp versions of CSS and JS files will be stored in these folders:

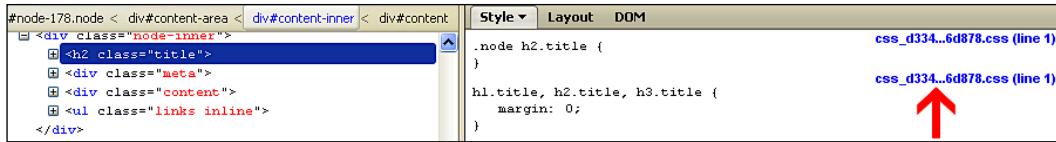


You can see this in action if you download and install the Firebug web developer's extension tool for Firefox and inspect your CSS. Using Firebug if I click to inspect the .node h2.title div class element in my Firebug HTML mode, notice that my style element is being pulled from a temp CSS cached file in the following location: http://variantcube.com/fire/files/css/css_e89481daee17b96def6fa7826e52a571.css.

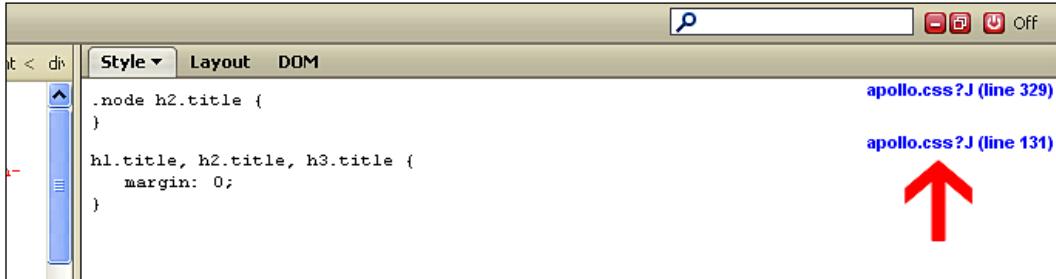
So, the cached CSS file is being saved to my /files/css folder. Cached versions of CSS files are saved in a css sub folder of your main site /files folder. I could potentially edit this CSS file, but the better practice in development would be to disable CSS optimization, and then edit my main theme's `apollo.css` file. Once you disable CSS optimization, and refresh your site and then access Firebug, you should see the following as the main style path:

<http://variantcube.com/fire/sites/all/themes/zen/apollo/apollo.css?9>

In this case, the CSS file is being loaded from your theme's root directory. It's not being cached now. Notice that if you go ahead and disable your CSS caching and then enable it again, the next temp CSS file will have a different filename than the first one we loaded above because it will be a new cached file by Drupal. This will happen each time you re-enable your CSS cache. This is why it's a good idea to only edit your main `apollo.css` or `style.css` (it may be named `style.css` if you're using another theme) file if you are having difficulty locating the CSS file to use and the correct enabled version of that file. Here's what Firebug will look like when you have CSS optimization enabled. Notice the long CSS filename and location of the temp file.



Here's what you'll see when CSS optimization is disabled:



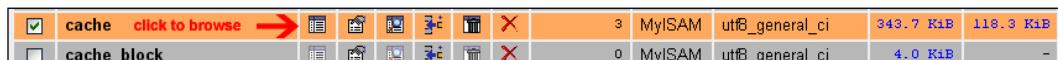
Cache tables in your MySQL database

If you view your MySQL Drupal database using a tool such as phpMyAdmin (provided your host provides this utility), then you can view your database tables including the cache tables. This will show you where the tables are located on your server and also give you a more detailed look at the content that is being cached. Follow these steps to view your database cache tables:

1. Open up your phpMyAdmin utility and select your Drupal site database in the left phpMyAdmin menu.
 2. The tables will load in the right screen.
 3. Look for the cache tables – this includes the following tables:

The phpMyAdmin table shows you the type of data in the table, the collation, and the size. For caching, it is necessary to pay special attention to the size of your database cache tables. As you cache more data, and if you do not have a minimum lifetime set on the cache, these tables will start to grow in data size. For example, currently the main cache table is 343.7 KiB in size.

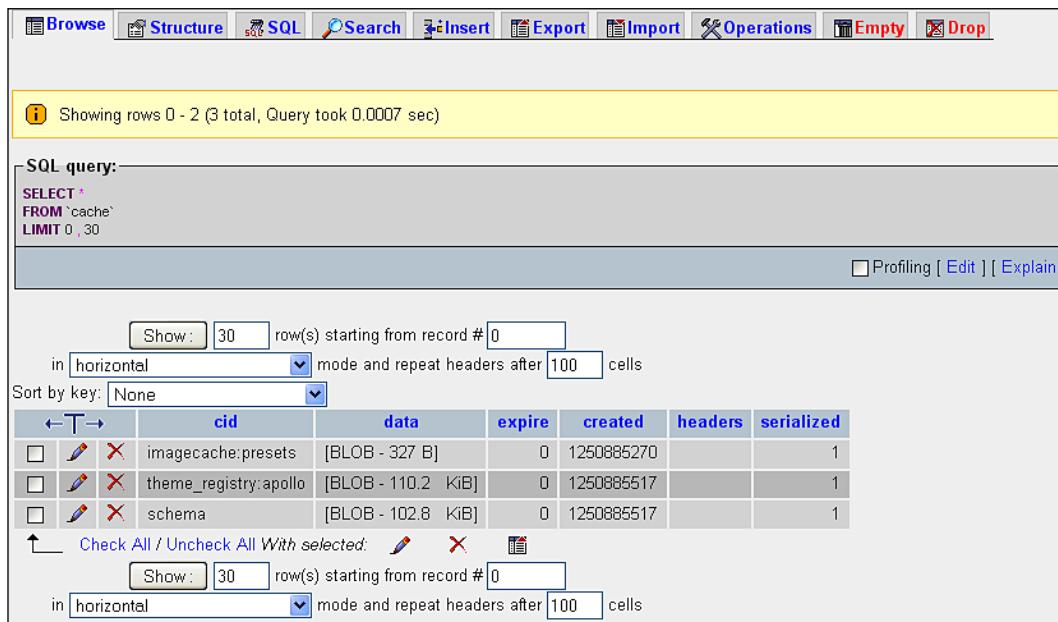
The main cache table contains your `imagecache` presets, if you have set these through the **Imagecache** module, and your theme registry files for your sub-theme. To access this data (browse the data), click on the filesize link in the size column next to the table in question or check the table box and then click on the **Browse** icon to browse the data.



A screenshot of the phpMyAdmin interface showing the 'cache' table. The table has two rows. The first row is selected (indicated by a checked checkbox) and has a 'Browse' button labeled 'click to browse'. The second row is not selected. The columns include: a checkbox column, the table name 'cache', a 'Browse' button, and columns for 'Rows', 'Type', 'Collation', 'Avg Row Length', 'Data Length', and 'Index Length'. The 'cache' table has 3 rows, is of type MyISAM, and has a total data length of 343.7 KiB and index length of 118.3 KiB.

<input checked="" type="checkbox"/>	cache	click to browse	3	MyISAM	utf8_general_ci	343.7 KiB	118.3 KiB
<input type="checkbox"/>	cache_block		0	MyISAM	utf8_general_ci	4.0 KiB	-

You will then see something like this, if you are browsing the main cache table:



A screenshot of the phpMyAdmin 'cache' table browse view. The top navigation bar includes 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Operations', 'Empty', and 'Drop'. A message indicates 'Showing rows 0 - 2 (3 total, Query took 0.0007 sec)'. The SQL query shown is: `SELECT * FROM `cache` LIMIT 0 , 30`. Below the query, there are options to 'Show: 30 row(s) starting from record # 0' and 'in horizontal mode and repeat headers after 100 cells'. The table itself has columns: cid, data, expire, created, headers, and serialized. Three rows are listed: 'imagecache:presets' (data: [BLOB - 327 B], expire: 0, created: 1250885270, headers: 1), 'theme_registry:apollo' (data: [BLOB - 110.2 KiB], expire: 0, created: 1250885517, headers: 1), and 'schema' (data: [BLOB - 102.8 KiB], expire: 0, created: 1250885517, headers: 1). There are also buttons for 'Check All / Uncheck All With selected' and additional 'Show' and 'in horizontal' options.

Let's look more closely at the `cache_page` table. This is the table that stores your cached Drupal node data. So, each time you load a Drupal node, the cached version (if you have Page caching enabled) will be stored in the `data` field of this table. If you refresh nodes on your site, load a bunch of Drupal nodes, and browse to this table to view the cached data, then you'll see something like this. For example, I went ahead and loaded this node numerous times using the web browser:
<http://variantcube.com/fire/node-178-book>.

Now when I browse to my `cache_page` table to browse the data stored, I see the following:

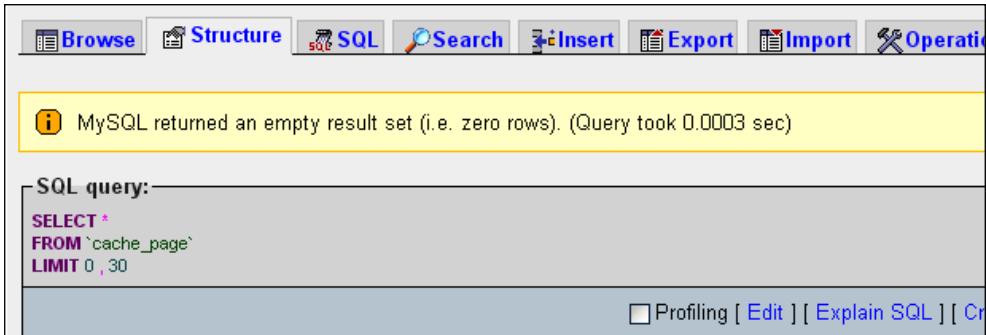
<input type="button" value="Show: 30 row(s) starting from record # 0"/> in horizontal mode and repeat headers after 100 cells Sort by key: None						
	cid	data	expire	created	headers	serialized
<input type="checkbox"/>	http://variantcube.com/fire/	[BLOB - 5.2 Kib]	-1	1250886237	Content-Type: text/html; charset=utf-8	0
<input type="checkbox"/>	http://variantcube.com/fire/node-178-book	[BLOB - 3.7 Kib]	-1	1250886242	Content-Type: text/html; charset=utf-8	0
<input type="checkbox"/>	http://variantcube.com/fire/node-177-book	[BLOB - 3.5 Kib]	-1	1250886250	Content-Type: text/html; charset=utf-8	0
<input type="checkbox"/>	http://variantcube.com/fire/node-189-forum	[BLOB - 3.6 Kib]	-1	1250886278	Content-Type: text/html; charset=utf-8	0

This is showing me that 4 nodes on the site have been cached so far – the main home page, **node-178**, **node-177**, and **node-189**. It also tells me the Drupal content type that the node is associated with. For example, **node-178** is a **book** and **node-189** is a **forum** posting. The table also gives us a column showing **data** size (per node) and when the cache is set to **expire**.

Clearing your performance cache

To clear your cache manually through the Drupal performance administration page go to **Administer | Site configuration | Performance**. Scroll to the bottom of the page and click on the **Clear cached data** button. This will clear your database cache tables. For example, if you click the button to clear the cache and then refresh your database `cache_page` table, you'll see that all of the previous primary key (`cid`) rows that were listed (and that we covered above) have been deleted.

Go ahead and try this! Clear your cache by means of the Drupal admin page and then refresh your database table view through phpMyAdmin to see the immediate effect. phpMyAdmin will give me the following result if I try browsing the `cache_page` table: MySQL returned an empty result set (i.e. zero rows).



The screenshot shows the phpMyAdmin interface with the following details:

- Navigation Bar:** Includes tabs for Browse, Structure, SQL, Search, Insert, Export, Import, and Operations.
- Status Bar:** Shows a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 sec)".
- SQL Query Field:** Displays the query: "SELECT * FROM `cache_page` LIMIT 0 , 30".
- Bottom Buttons:** Includes Profiling, Edit, Explain SQL, and Create buttons.

Clearing your theme registry

Another component of Drupal caching is the theme registry. Your theme and its associated functions and CSS elements can be cached by Drupal as it loads. Drupal stores this information as cached data. You may only need to worry about the theme registry if you are developing your theme and adding theme override functions to your `template.php` files and other Drupal theme template files.

Drupal template files control all of the PHP and HTML output of your Drupal theme. These template files are stored in the root folder of the theme you are using. You will see the following template files (and you may see more depending on how complex the theme is) when you look in a theme folder: `template.php`, `page.tpl.php`, `node.tpl.php`.

After adding a function to your template files, you may notice that your changes do not take effect immediately when you refresh your page. You can do the following to ensure that your function overrides show up immediately.

Clear your main Drupal cache using the method discussed earlier. Clear your cache at **Administer | Site configuration | Performance**. This will rebuild the theme registry on your site. Then reload the page in question and you should see your TPL changes. Another method of clearing the theme registry is to visit your **Site building | Themes configuration page** and click on the **Save configuration** button to re-save your theme's configuration. By doing this you will clear the theme registry and your latest theme changes and tweaks at the template level will be visible.

Enable your theme registry through your theme admin, if the theme developer provides an admin checkbox for this API function. For example, we're using a sub-theme of the Zen theme. As we're using the Zen theme, we have this additional **Theme registry** checkbox available to us. This is not included in all contributed themes.

If you go to your theme configuration page at **Administer | Site building | Themes** and then click on **Configure**, and select your specific sub-theme, you'll see a **Theme Registry** section. In this case with Zen, it's in a pane of the sub-theme configuration page, titled **Theme development settings**.

Check the box to enable the rebuilding of the theme registry on every page during development.

Drupal warns you not to enable the theme registry on a production website, as it can cause performance issues if you have it enabled and are rebuilding the theme registry on every page load. So, for now we're going to leave it disabled by unchecking the box. You can then select the box again later if you start doing theme development.

Theme development settings

Theme registry: Rebuild theme registry on every page.
During theme development, it can be very useful to continuously [rebuild the theme registry](#). WARNING: this is a huge performance penalty and must be turned off on production websites.

Wireframes: Display borders around main layout elements
[Wireframes](#) are useful when prototyping a website.

For more on the Drupal theme registry click on the link in the text next to the theme registry checkbox that is hyperlinked to **Rebuild the theme registry on every page**. This will launch a help page on drupal.org with more information about theme registry functionality: <http://drupal.org/node/173880#theme-registry>.

Running cron manually

Another tip is to run your `cron` task often in order to clear out your database of stale data including old log entries (saved through the Drupal **watchdog** module), cache entries that have not been cleared when you cleared your cache tables, and any other stale data. The `cron` task will also activate other maintenance tasks on your site and force these tasks to run, including re-indexing your site for the Drupal search functionality, force RSS feeds to refresh with new feed content and to update and perform various tasks that are dependent on this utility. For example, if you have the **Statistics** module enabled, you'll need to run `cron` in order to index your latest page hits and views. You will also need to run `cron` in order to activate the **Search** module indexing of your site's new content.

To run `cron` you tell Drupal when to execute the `cron.php` script that sits at the root of your site's directory. `cron` will then run on a specified timetable—either immediately if you run it manually or on a scheduled time specified in hours, day of the month, or month.

The `cron.php` file is installed with Drupal core when you extract your Drupal directory to your server. You just need to tell Drupal when to execute the script. When it is run, the `cron.php` file will load at the following URL of our site:

<http://variantcube.com/fire/cron.php>

You can execute the cron script by going to that URL. You can also run it through the Drupal Status report by following these steps:

1. Go to **Administer | Reports | Status report**.
2. Look for the table row for cron maintenance tasks. This will show you when the cron was last run. In our case, we just ran it through the script URL, so it's showing us that it was **Last run 1 sec ago**.
3. Go ahead and click on the **run cron manually** link.
4. This will run cron again and you'll receive a green message when the page reloads telling you that cron ran successfully.

The screenshot shows the 'Status report' page from a Drupal site. At the top, there is a breadcrumb navigation: Home > Administer > Reports. Below the header, the title 'Status report' is displayed. A green box contains the message 'Cron ran successfully.' Underneath, a text block says: 'Here you can find a short overview of your site's parameters as well as any problems detected with your installation. It may be useful to copy and paste this information into support requests filed on drupal.org's support forums and project issue queues.' A table follows, showing site parameters and their status. The table has two columns: 'Parameter' and 'Status'. The parameters listed are: Drupal (version 6.13), Access to update.php (Protected), CTools CSS Cache (Exists), Configuration file (Protected), and Cron maintenance tasks (Last run 1 sec ago). The last row of the table contains the text 'You can run cron manually.' in blue.

Drupal	6.13
✓ Access to update.php	Protected
✓ CTools CSS Cache	Exists
✓ Configuration file	Protected
✓ Cron maintenance tasks	Last run 1 sec ago
You can run cron manually.	

Installing the Poormanscron module

It will become inconvenient to have to login to your site and run cron manually each time you want to clear stale data, re-index your search module, and perform other routine tasks. If you do not have access to set up a scheduled cron task on your server via cPanel or through other methods your host provides, you may want to consider installing the Drupal **Poormanscron** contributed module. This allows you to install a cron module and configure it to run scheduled cron tasks. The module's project page is here:

<http://drupal.org/project/poormanscron>

Poormanscron

[View](#) [CVS instructions](#)

Uwe Hermann - September 28, 2003 - 16:45 [Administration](#) · [Modules](#)

A module which runs the Drupal cron operations without needing the cron application.

For every page view, this module checks to see if the last cron run was more than 1 hour ago (this period is configurable). If so, the cron hooks are executed, and Drupal is happy. These cron hooks fire after all HTML is returned to the browser, so the user who kicks off the cron jobs should not notice any delay.

Releases

Official releases	Date	Size	Links	Status
6.x-1.0	2008-Feb-06	11.33 KB	Download · Release notes	Recommended for 6.x ✓
5.x-1.1	2007-Jan-20	11.39 KB	Download · Release notes	Recommended for 5.x ✓
4.7.x-1.0	2007-Jan-14	10.38 KB	Download · Release notes	Recommended for 4.7.x ✓

Let's go ahead and install the module. Here are the steps:

1. Download the **6.x-1.0** version from the project page.
2. Extract the folder to your desktop and then upload to your `/sites/all/modules` folder via SFTP.
3. Enable the module through your admin modules list.

Poormanscron 6.x-1.0 Internal scheduler for users without a cron application.

Now let's configure the module. Here are the steps:

1. Go to **Site Configuration | Poormanscron**.
2. Here you can set your cron run intervals. We'll set the cron to run every 60 minutes. Every hour, the cron tasks will execute after the first page loads immediately at the 60 minute mark.
3. We'll set the error wait setting to 10 minutes. This means that if the cron run experiences an error and does not run completely, the task will re-run and try again 10 minutes later.
4. We'll log successful cron runs to the watchdog table in our database and we'll see these appear as line items in our **Recent log entries** report.

5. You can also log the entire cron progress report to your recent log entries.
If you do not want to see a long list of messages related to the cron run, just leave this set to **No**.

Time intervals

Cron runs interval:
 Minimum number of minutes between cron runs. Cron will actually execute during the first page request after the interval has elapsed.

Retry interval:
 The number of minutes to wait after a cron run error before retrying.

Logging

Log successful cron runs:
 Yes If you want to log successful cron runs to the Drupal watchdog, say Yes here. If those messages annoy you, disable them by selecting No.

Log poormanscron progress:
 No If you want to log the progress of a poormanscron run to the Drupal watchdog, say Yes here. If those messages annoy you, disable them by selecting No.



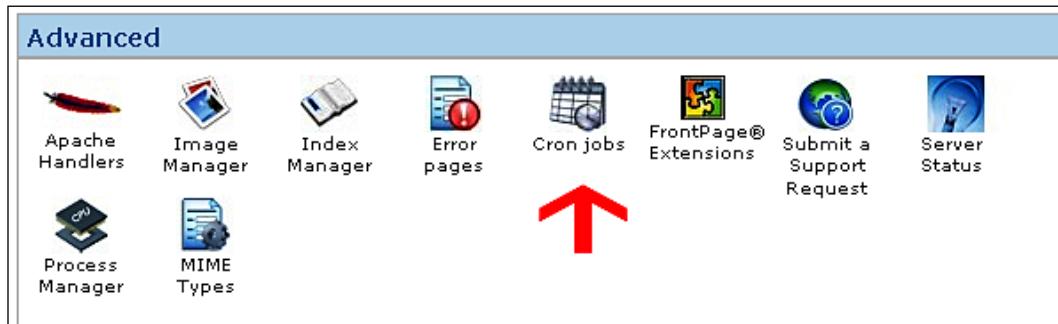
It appears that developers and Drupal webmasters are divided on whether this module contributes to other performance issues on your site. For example, it's been reported that the Drupal system needs to check through the module if it's time to run the cron job. This check can take time and cause potential performance bottlenecks. There is more about the potential disadvantages of using the **Poormanscron** module in this article: <http://drupal.org/node/102647>.

The advantages of using the module for those of us who do not have access to set up cron scripts on our server are large. We can rely on this module instead of having to login to our site every day or every hour to run our cron manually.

Setting up cron through cPanel

If you have a cPanel or Plesk utility for your server, you can also set up and configure cron tasks using this tool. We're going to login to our cPanel now and take a look at the interface to set up cron tasks.

1. Login to cPanel.
2. Look for a cron jobs icon/button under the **Advanced** section of your cPanel utility. Click on this icon.



3. cPanel will give you some introductory text detailing what cron is and then allow you to use either a standard method of setting the cron or the Advanced Unix style of configuration. More about this is detailed in the following article on drupal.org: <http://drupal.org/node/369267>.

Cron Jobs

Cron jobs allow you to automate certain commands or scripts on your site. You can set a command or script to run at a specific time every day, week, etc. For example, you could set a cron job to delete temporary files every week so that your disk space is not being used up by those files.

Warning: You need to have a good knowledge of Linux commands before you can use cron jobs effectively. Check your script with your hosting administrator before adding a cron job.

Please choose your experience level:

Backing up your site using SFTP/FTP and cPanel

Now that we have completed our initial investigation of the best practice methods for maintaining and monitoring our new Drupal 6.13 site, it's a good idea to run a full backup of the site directory (all of our Drupal files and folders), and of the database. We did this in Chapter 1 before we completed our initial Drupal 5.x upgrade, but we're now going to look at this in more detail. Backups are essential to keep our site performing well over time. Eventually, you may run into an issue or glitch in your site or database, which will cause you to be forced to restore some of your Drupal core or module files and restore your Drupal database. You may never need to resort to a backup, but it's a requirement to have a backup in case you do need it because issues and errors in your site and database can come up unexpectedly.

The first method is to run a backup of all of your Drupal files and folders. One method of doing this is through SFTP. Connect to your site using an SFTP client and then copy all of your folders and files back to your local desktop or local server/sandbox environment. Be sure to run this backup through secure FTP, if your host gives you access to an SFTP client or connection. This will minimize the risk of having your data comprised as it is being transferred via FTP. This is the method of backup we ran in Chapter 1.

Another method of creating a backup of your site and/or server is to use a tool that your cPanel may provide. Utilities, such as cPanel, often provide a backup interface that you can use to create a full backup of your site's home directory or `public_html` directory. You can then save this backup to another directory on your server and restore from this directory if necessary. The benefit of doing a backup using this method is that it's potentially more secure than your SFTP method, as the data is staying on your remote server. Here are the steps:

1. Login to your cPanel.
2. Locate the **Backups** icon/button usually under a section titled **Files** or **File Management**. Click on that icon.



- This will launch a page that will give you options to download either a **Full Backup** (all of your site directories, databases, and e-mail forwarders and filters), or a specific **Home Directory Backup**, which will include all the site directories in the home directory of your web server.

Full Backup
A full backup includes all of the files in your home directory, your MySQL Databases, and your email forwarders and filters. You can backup your account to preserve your data or use a backup file to move your account to another cPanel server.

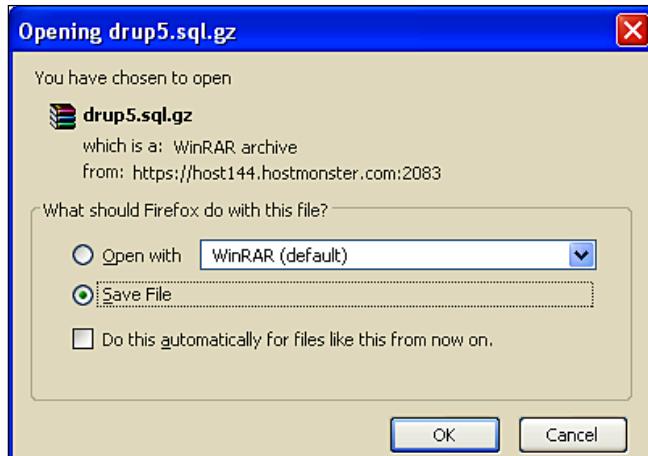
[Download or Generate a Full Web Site Backup](#)

System Backups
 Select a system backup to download:

Partial Backups

<p>Download a Home Directory Backup</p> <p>Home Directory</p>	<p>Restore a Home Directory Backup</p> <p><input type="text"/> Browse... Upload</p>														
<p>Download a MySQL Database Backup</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #0070C0; color: white;">DATABASES</th> </tr> </thead> <tbody> <tr><td>variantc_adsw60</td></tr> <tr><td>variantc_adsw60civi</td></tr> <tr><td>variantc_adsw60civi2</td></tr> <tr><td>variantc_aol</td></tr> <tr><td>variantc_arrayloop</td></tr> <tr><td>variantc_beaver</td></tr> <tr><td>variantc_blonder</td></tr> <tr><td>variantc_crt1</td></tr> <tr><td>variantc_civicrm</td></tr> <tr><td>variantc_civicrm2</td></tr> <tr><td>variantc_drpl1</td></tr> <tr><td>variantc_drpl2</td></tr> <tr><td>variantc_drup5</td></tr> </tbody> </table>		DATABASES	variantc_adsw60	variantc_adsw60civi	variantc_adsw60civi2	variantc_aol	variantc_arrayloop	variantc_beaver	variantc_blonder	variantc_crt1	variantc_civicrm	variantc_civicrm2	variantc_drpl1	variantc_drpl2	variantc_drup5
DATABASES															
variantc_adsw60															
variantc_adsw60civi															
variantc_adsw60civi2															
variantc_aol															
variantc_arrayloop															
variantc_beaver															
variantc_blonder															
variantc_crt1															
variantc_civicrm															
variantc_civicrm2															
variantc_drpl1															
variantc_drpl2															
variantc_drup5															
<p>Restore a MySQL Database</p> <p><input type="text"/> Browse... Upload</p>															

4. You can also download backups of your MySQL databases through this admin page. Click on the links to the database in question to download its backup. For example, here we'll click on the link to the **variantc_drup5** database.



5. You can also restore your home directory backup and your MySQL databases from this page.

You can also use the Backup wizard option through cPanel, if your cPanel provides this. It's basically the same functionality as described above with the Backups utility, but this walks you through the steps, one at a time, in wizard workflow format. The wizard asks you if you want to **Backup/Restore** or **Restore**. Then you click and follow the steps. If you select a Backup, it will then prompt you for whether you want to do a **Full Backup** or a **Partial Backup**.

A screenshot of the hostmonster Backup Wizard interface. The top navigation bar includes "cPanel", "Domain Manager", "Upgrades", "Postini", "Dedicated IP", "SSL Certificates", "Profile/Billing", "Checkout", "Logout", and "HELP". The main area is titled "Backup Wizard" with "Steps:" below it. Step 1 is highlighted with a circle and labeled "Backup/Restore". Step 2 is labeled "Full or Partial Backup" and Step 3 is labeled "Download".
Backup:
This feature allows you to download a zipped copy of your entire site or parts of it onto your computer.
The following are backed up and included in a zip file for your convenience:
Home Directory
MySQL Databases
Email forwarders configuration
Email filters configuration
Backup →
Restore:
This feature allows you to restore parts of your cPanel by uploading your partial backup zip file(s) that you downloaded from the backup feature.
The following can be restored:
Home Directory
MySQL Databases
Email forwarders configuration
Email filters configuration
Restore →

Backing up your database through phpMyAdmin

You can do a full backup of your database using phpMyAdmin, if your host provides you access to this utility.

1. Launch phpMyAdmin from your cPanel.



2. You will be redirected to your phpMyAdmin URL.
3. When it loads, locate your database name in the left menu and click on it.
4. Once your database tables load, click on the **Export** tab.
5. Select the tables you want to export or **Select All** to export the entire database. Make sure you're exporting the structure and the data of the database, so check to make sure those checkboxes are selected. You can also choose to only export the structure of the database or just export the data.
6. Make sure to select the **SQL** radio button under the **Export** table in the left column in order to export the data as SQL data.

7. Check the **Save as** file checkbox and then click on **Go** to initiate the export.

The screenshot shows the 'Export' section of the phpMyAdmin interface. On the left, a tree view lists tables: access, accesslog, actions, actions_aid, advanced_help_index, authmep. To the right, the 'Options' panel is expanded, containing sections for 'Structure' (checkboxes for Add custom comment, Enclose export in a transaction, Disable foreign key checks, SQL compatibility mode set to 'NONE'), 'Add into comments' (checkbox for Creation/Update/Check dates), and 'Data' (checkboxes for Complete inserts, Extended inserts, Maximal length of created query set to 50000, Use delayed inserts, Use ignore inserts, Use hexadecimal for BLOB, Export type set to 'INSERT'). At the bottom, the 'Save as file' section is checked, with 'File name template' set to 'DB' and 'Compression' set to 'None'. A 'remember template' checkbox is also present. A 'Go' button is at the bottom right.

Tweaking your HTACCESS file

Your site directory contains an `.htaccess` file that sits at the root level of your Drupal site. This file comes packaged with the Drupal install and gets installed with Drupal by default. You can view and edit this file using your cPanel or SFTP, if you first change its permissions to write permissions as it's a ready-only file.

Your `.htaccess` file allows you to tweak directory permissions, specify how Drupal handles error messages, and present errors (404 errors) to the end user. You can also override your PHP settings in this file in a similar way to how you override your PHP settings in your `settings.php` file and your `php.ini` file. This presents another option. Usually host servers provide multiple methods of tweaking your PHP settings in case you do not have permissions or access as a developer to the Drupal configuration file, the `php.ini` file, or the `.htaccess` file. In your `.htaccess` file, you'll look for the version of PHP you're using (in this case PHP 5) and then look

for the lines of `php_value` code that's under this version. For example, this is what that set of values looks like in our `.htaccess` file:

```
# PHP 5, Apache 1 and 2.  
<IfModule mod_php5.c>  
php_value magic_quotes_gpc          0  
php_value register_globals         0  
php_value session.auto_start       0  
php_value mbstring.http_input      pass  
php_value mbstring.http_output     pass  
php_value mbstring.encoding_translation 0  
</IfModule>
```

You can add overrides for `php_memory_limit` and `upload_max_size` here to the `.htaccess` file and then save your file. Then refresh your Status report and you should see your changes in effect. Let's go ahead and add a new `upload_max_size` value here, as we want to increase our local value above the current 2 MB default.

To do this we're going to add the following lines of code to the PHP5 section of our `.htaccess` file:

```
php_value post_max_size 20M  
php_value upload_max_filesize 10M
```

1. Change your permissions through cPanel on your `.htaccess` file to 777 so that you can write to the file.
2. Edit the file.
3. Add the 2 lines of code mentioned above and make the `post_max_size` 20M. Make the `upload_max_size` 10M.
4. Save changes to your file.
5. Refresh your Status report through Drupal. If for some reason your changes did not take immediate effect, you may need to ask your hosting service to restart the Apache web server.

6. Your .htaccess file code should now look similar to this:

```
# PHP 5, Apache 1 and 2.  
<IfModule mod_php5.c>  
    php_value magic_quotes_gpc          0  
    php_value register_globals         0  
    php_value session.auto_start      0  
    php_value mbstring.http_input     pass  
    php_value mbstring.http_output    pass  
    php_value mbstring.encoding_translation 0  
    php_value post_max_size          20M  
    php_value upload_max_filesize    10M  
</IfModule>
```

Summary

Congratulations! Here's a recap of what you accomplished in this chapter and why it's important:

- You reviewed your main `settings.php` file in detail and made tweaks to the `base_url` path designation, and you learned how to tweak PHP settings in your `settings.php` file and your `.htaccess` file.
- You reviewed your contributed modules and themes and deleted any of these that you no longer needed enabled, and also deleted any files in your Drupal directory that you no longer needed on your production site.
- You enabled the Drupal 6.x Update Status module that is now part of core Drupal.
- You learned how to run your `cron` tasks through the Drupal admin and cPanel, and you installed and configured the Poormanscron module to set up and configure your `cron`.
- You received an introduction to Drupal caching and learned where cached data is stored and why it's stored in your Drupal database. You learned how to maintain this cached data, and how to clear it and enable it.
- You learned the best practice methods for backing up your Drupal site directory and database.

You're now ready to do more advanced monitoring of your site using some of Drupal's contributed modules including the **Devel (Development)** module, as well as take a closer look at how Drupal caching works, and monitor your log entries and errors. See you in Chapter 3!

3

Using Development Modules and Tools

In this chapter, we're going to use the Devel module to monitor performance on our Drupal site to assist us in locating bottlenecks. The Devel module can be used to quickly gather detailed information on your site's theme elements, monitor your site's page loads and database queries, generate test content for your site, and give you easy methods for clearing your site's theme registry and performance cache. It's an essential tool for the performance-minded Drupal developer. We're also going to look more closely at how your site logs access entries and how you can view your log entries to troubleshoot performance issues.

Our focus in this chapter will be on using the Devel module to monitor performance. The module logs performance issues, such as page load times and memory usage, which allows you as a developer to find issues quickly and troubleshoot. In this chapter, we'll learn how to enable detailed and summarized performance logging on our site.

To summarize, here's what we'll be doing in this chapter:

- Viewing log entries and reports
- Using the Devel module to monitor site performance
- Using the Development block to access site performance logs and to tweak functionality

Viewing and inspecting recent log entries

The Drupal **Dblog** (formerly called **Watchdog** in earlier Drupal versions) module keeps recent log entries in your Drupal administration as long as you have not cleared your performance cache. You can tell Drupal how long you want to keep the log entries stored in your site's database.

Viewing your recent log entries

Having saved the configuration settings, launch the **Recent log entries** report pages by going to **Administer | Reports | Recent log entries**. This will launch the following page: <http://variantcube.com/fire/admin/reports/dblog>

The logs will be displayed in a table by **Type**, **Date**, and the detailed log **Message** or error. Errors will be noted with a red X icon, and warnings will be flagged with a yellow exclamation point icon. For example, a **php** error will usually throw a red X line item, while a **page not found** error will show a yellow exclamation point warning. A column is also presented in the log entries table showing the user account and session during which the error occurred.

You should check this report often so that you can confirm that your site is logging errors and which errors need troubleshooting. Often, if you are getting an error on your site it's a good idea to immediately try running your recent log entry report and check to see if the error is being logged. Viewing the log entries may provide you with the code line in a specific module and the location in the module code of the specific error. This can greatly reduce the time spent searching for the issue, as the error log will usually narrow the error down to a specific line in the template code. The logs also tell you the date and time of the error so that you can locate an issue that occurred at a specific moment on the site and/or server.

Notice in this screenshot the Dblog is reporting three **php** errors from August 18. The error messages are hyperlinked and if we click on the links, we can see more details of the entire error.

Type	Date▼	Message	User	Operations
user	08/18/2009 - 19:12	Session opened for admin.	admin	
user	08/18/2009 - 16:44	Session closed for admin.	admin	
cron	08/18/2009 - 16:34	Cron run completed.	admin	
✖ php	08/18/2009 - 16:26	Table 'cache_views_data' already exists query: CREATE TABLE cache_views_data (`cid` VARCHAR(255) NOT NULL DEFAULT '', `data` LONGBLOB DEFAULT NULL, `expire` INT NOT NULL DEFAULT 0, `created` INT NOT NULL DEFAULT 0, `headers` TEXT DEFAULT NULL, `serialized` SMALLINT NOT NULL DEFAULT 1, PRIMARY KEY (cid), INDEX expire (expire)) /*!40100 DEFAULT CHARACTER SET UTF8 */ in /home/variantc/public_html/fire/includes/database.inc on line 517.	admin	
✖ php	08/18/2009 - 16:26	Duplicate key name ...	admin	
✖ php	08/18/2009 - 16:25	Invalid argument supplied for foreach() in ...	admin	
content	08/18/2009 - 16:25	Updating widget type ...	admin	
content	08/18/2009 - 16:25	Updating widget type userreference_buttons ...	admin	

For example, the first error in the list is related to the update.php script being run. Drupal is telling us that the table the update.php script is trying to create—cache_views_data—already exists.

Type	php
Date	Tuesday, August 18, 2009 - 16:26
User	admin
Location	http://variantcube.com/fire/update.php?id=2&op=do
Referrer	http://variantcube.com/fire/update.php?op=start&id=2
Message	Table 'cache_views_data' already exists query: CREATE TABLE cache_views_data (`cid` VARCHAR(255) NOT NULL DEFAULT '', `data` LONGBLOB DEFAULT NULL, `expire` INT NOT NULL DEFAULT 0, `created` INT NOT NULL DEFAULT 0, `headers` TEXT DEFAULT NULL, `serialized` SMALLINT NOT NULL DEFAULT 1, PRIMARY KEY (cid), INDEX expire (expire)) /*!40100 DEFAULT CHARACTER SET UTF8 */ in /home/variantc/public_html/fire/includes/database.inc on line 517.
Severity	error
Hostname	72.236.192.234
Operations	

Remember to run cron to clear your older log entries and keep the log entries screen up-to-date.

Logging and alerts configuration

First, let's check our logging configuration and settings. To access your **Dblog** configuration follow these steps:

1. Go to **Site configuration | Logging and alerts**: <http://variantcube.com/fire/admin/settings/logging>

Database logging

Settings for logging to the Drupal database logs. This is the most common method for small to medium sites on shared hosting. The logs are viewable from the admin pages.

2. Click on **Database logging**.
3. This will load the configuration page where you can tell Dblog the maximum number of log entries to keep in the database. So, if you select to keep **1000** log entries at any given time, it will be the most recent **1000** log entries. When you run cron, the older entries will be cleared and the log will be refreshed.
4. Click on **Save configuration**.

Discard log entries above the following row limit:

The maximum number of rows to keep in the database log. Older entries will be automatically discarded. (Requires a correctly configured [cron maintenance task](#).)

Page not found and access denied errors

The Dblog module also keeps tabs on your top 'access denied' (403) errors and top 'page not found' (404) errors. For access denied errors go to **Reports | Top 'access denied' errors** at <http://variantcube.com/fire/admin/reports/access-denied>. This will display a table that shows the Drupal page path that generated the access denied error and how many times it's been generated. In the following screenshot, you see that the /admin path has generated access denied errors three times and the /logout path has generated an error once. Keeping an eye on this it can also tell you

where a user interface bottleneck might be occurring. For example, if you see a lot of access denied errors on a specific page, there may be an issue on your site where you have a menu link going to this node that anonymous users are clicking on.

Home > Administer > Reports	
Top 'access denied' errors	
Count▼	Message
6	logout
3	admin

Here's a screenshot of the **Top 'page not found' errors**. As with the **Top 'access denied' errors**, this table shows the number of times the page generated the 404 error. This can also help you to determine why a page does not exist on your site, but continues to be hit, and makes it easier to create a page that does function on your site.

Home > Administer > Reports	
Top 'page not found' errors	
Count▼	Message
3	home
1	update.php
1	panel_page_1
1	files/css/css_7b3560b393f129818bfd483602e13ad0.css
1	files/js/js_8f87a55059a3c34fa1049b95b0f5d18b.js
1	use

The recent log entries reports are invaluable to you as a performance-minded developer. If you check them regularly and try to keep them as clear of errors and warnings as possible, then your site will continue to perform well.

The Devel module

The Devel module provides a suite of tools for the Drupal theme and performance developer. Two major components of the Devel module include helper utilities that monitor performance of your site, and tools for themers that allow for quick and more accurate theme development.

Developers can use the module to monitor all of the database queries that occur as each Drupal page loads. This tool will show you how many times the database query executes on each page when it loads. Knowing this is essential when monitoring performance because it allows you to find out which queries are repetitive and creating potential bottlenecks or other performance issues especially on a large site or multisite environment. As the module project page points out, you do not want to see queries running multiple times on the same page. The development component of the module also tells you how long the queries take to run. This can give you detailed information on which queries and functions may need to be tweaked and optimized during your development work.

You can also run `print_r` commands to print your arrays. This is helpful to a Drupal developer, as it will show you all of the array variables available and you can do it through the module interface instead of having to run a `print_r` command on each Drupal node or in your template files.

The theming component of Devel offers the developer various tools that are similar in scope and functionality to the Firebug extension or the Firefox Web Developer extension. This has caused the Devel module to garner the nickname 'firebug for Drupal'. The theme wizard in Devel gives you a visual outline of your template functions and available variables, and allows you to select an element on your Drupal site visually to gather this information. So, you can outline an element such that a navigation menu block and then view the theme information on the Devel screen.

You can also use the Devel module to quickly generate test content, what's commonly called dummy `lorem ipsum` content and test nodes, and content types for your site. The module also gives you a report of any node access issues your site may be experiencing.

The Devel module is available for download from its project page at [drupal.org](http://drupal.org/project/devel) (<http://drupal.org/project/devel>). I encourage you to read all of the information on the project page and to browse the issues and forum support tickets that have been generated for this module, as this research will teach you more about how it works.

The current version we are using is 6.x-1.17. Let's go ahead and download the latest version and upload it to our site. On our site we'll actually be upgrading the 1.16 version that we've had installed since our 6.x upgrade.

The screenshot shows the Devel module page for the theme_username() function. At the top, there are links for 'View' and 'CVS instructions'. Below that, the author is listed as 'moshe weitzman - September 28, 2003 - 16:44' and the status is 'Modules · Developer · Drush'. A prominent message reads '#D7CX: I pledge that Devel will have a full Drupal 7 release on the day that Drupal 7 is released.' The main content area is titled 'Devel' and contains a bulleted list of features. To the right, a detailed call stack is displayed under the heading 'Drupal Themer Information'. It shows the function being called is 'theme_username()' and lists candidate function names: minnelli_username, phptemplate_username, garland_username, and theme_username. The duration of the function execution is 0.6 ms. Below this, a 'Function Arguments' section shows an array with one element, containing four items: nid (String, 2 characters) 25, type (String, 5 characters) story, and language (String, 2 characters) en.

Devel

View **CVS instructions**

moshe weitzman - September 28, 2003 - 16:44

Modules · Developer · Drush

#D7CX: I pledge that Devel will have a full Drupal 7 release on the day that Drupal 7 is released.

A suite of modules containing fun for both module and theme developers ...

Devel

- helper functions for Drupal developers and inquisitive admins. This module can print a summary of all database queries for each page request at the bottom of each page. The summary includes how many times each query was executed on a page (shouldn't run same query multiple times), and how long each query took (short is good - use cache for complex queries).
- Also a `dprint_r($array)` function is provided, which pretty prints arrays. Useful during development. Similarly, a `ddebug_backtrace()` is offered.
- much more

Theme developer (Drupal6 only)

Drupal Themer Information

Parents: `phptemplate_node_submitted < node.tpl.php < page.tpl.php`

Function called:
theme_username()

Candidate function names:
minnelli_username < phptemplate_username < garland_username < theme_username

Duration: 0.6 ms

Function Arguments

... (Array, 1 element)
0 (Object) stdClass
nid (String, 2 characters) 25
type (String, 5 characters) story
language (String, 2 characters) en

Installing and enabling Devel

Let's install the Devel module and proceed with our initial configuration of the module.

1. Download the latest release of Devel from its Drupal project page.
2. Extract it to your local desktop and upload the folder to your `/sites/all/modules` directory in your Drupal site using SFTP or cPanel.

3. Enable the module and its components via your modules admin page. Make sure to enable all modules including **Devel**, **Devel generate**, **Devel node access**, **Macro**, **Performance Logging**, and **Theme Developer**.

Development			
Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	Devel	6.x-1.17	Various blocks, pages, and functions for developers. Depends on: Menu (enabled) Required by: Theme developer (disabled)
<input checked="" type="checkbox"/>	Devel generate	6.x-1.17	Generate dummy users, nodes, and taxonomy terms.
<input checked="" type="checkbox"/>	Devel node access	6.x-1.17	Developer block and page illustrating relevant node_access records.
<input checked="" type="checkbox"/>	Macro	6.x-1.16	Allows administrators to record and playback form submissions.
<input checked="" type="checkbox"/>	Performance Logging	6.x-1.17	Logs detailed and/or summary page generation time and memory consumption for page
<input checked="" type="checkbox"/>	Theme developer	6.x-1.17	Essential theme API information for theme developers Depends on: Devel (disabled), Menu (enabled)

4. Save your module configuration.
5. Check your site Status report to see if you need to run `update.php` after enabling this module. The performance module components of Devel do have database tables they write to, so if you're upgrading Devel you may need to update your database schema as well.
6. Once saved, you should notice an immediate change to your site. In the lower left-hand corner of your site you'll see a small gray box with the title **Themer info** and a check box next to it.
7. Before we start using the **Themer info** functionality, let's set some Devel module permissions.

Checking Devel module permissions

Another item to check before you or any of your site admins start using the Devel module features and the Development block discussed later in this section are the permissions to use the module. Go to **User management | Permissions** and look for the Devel module permissions. Make sure you have checked the appropriate permissions for your site admins or other admin roles. This may include allowing them permissions to:

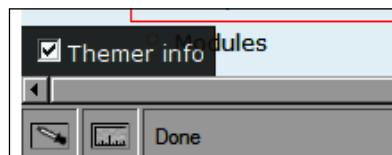
- Access Devel information
- Display source code
- Execute PHP code
- Switch users

You may also give them permissions to view **devel_node_access_information** as part of the **devel_node_access** module.

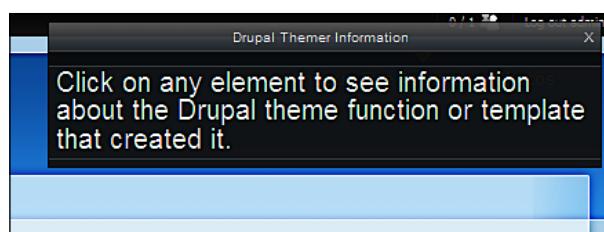
devel module		
access devel information	<input type="checkbox"/>	<input checked="" type="checkbox"/>
display source code	<input type="checkbox"/>	<input checked="" type="checkbox"/>
execute php code	<input type="checkbox"/>	<input checked="" type="checkbox"/>
switch users	<input type="checkbox"/>	<input checked="" type="checkbox"/>
devel_node_access module		
view devel_node_access information	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Enabling Themer info

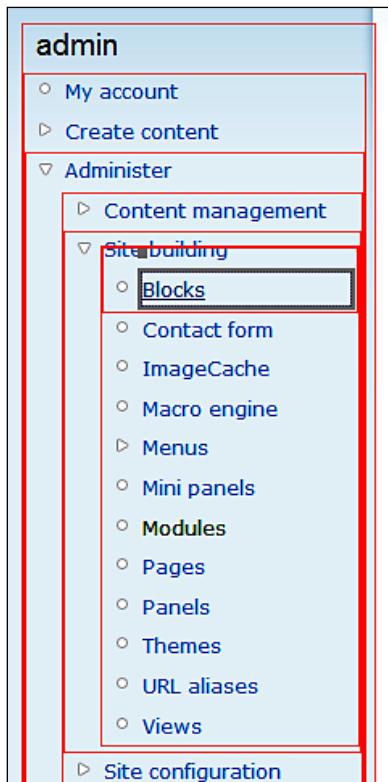
To enable the **Themer info** tool, click on the checkbox or anywhere in the gray rectangle.



Once clicked, you'll see a second rectangle appear in the upper right corner of your site. This box tells you to click on elements on your page to see detailed information about the Drupal theme function and information about the template file that contains the theme function.



As you move your mouse around the Drupal page you'll notice red borders surrounding any element that you move over. For example, as I move my mouse over the left admin menu sidebar, a red box surrounds the entire menu element. When I click on the element the red border box turns gray and the corresponding theme function information shows on my right-hand Devel box. Notice that you can drill down to the specific `` or `` element in the navigation and get information on those elements.



When you click on an element, the element is surrounded by a gray box or border. This helps to differentiate an active clicked item that will show you information in the Themer information box from other elements that you have rolled your mouse over.

So, in this case if I select the **Blocks** menu list item in my administration menu block, the Themer toolbox tells me the following information:

- The function called is `theme_menu_item_link()`. This is already very helpful to us if we're going to do any theming. If we wanted to override this function in our main `template.php` file, we already know the name of the function through the Devel module. We do not need to hunt for the function code in the actual core Block module files. This tool tells us the name of the function and now we can override that function in our specific theme template by replacing `theme_` with the name of our theme. So, if we're using the Garland theme, we'd replace `theme_` with `garland_`. The helper text in the theme developer box tells us that this is a potential candidate for the override.



The Themer information also tells you how long it takes the function to load in Drupal on that specific page. So, when this page loads, the function loads in **0.25 ms**. If you move your mouse over other page elements, you'll notice that the loading duration will change, increasing or decreasing depending on how complex the function is.

This is an introduction to how the Devel module can be used for theming purposes. Our focus is on performance, so let's enable some of the module's performance-based components.

Devel settings

To enable more Devel module settings, including performance options, go to **Site configuration | Devel settings** or to here: <http://variantcube.com/fire/admin/settings/devel>

This is where you can configure Devel to collect database query information and display query logs. Follow these steps to configure performance logging:

1. Check the box next to **Collect query info** if you want Devel to collect database query info.
2. Check the **Display query log** box if you want Devel to display a log of these database queries. The log will also display the execution time for each query. It will also keep tabs on repeated queries and note these in the logs in a separate # column. These queries will be highlighted in red to draw your attention to them and then you can decide whether you need to focus caching mechanisms on these specific queries to improve performance.
3. Tell Devel how you want to sort the query log—by source or by duration. We will leave sorting by **source** so that it displays them in the order of execution.
4. In the **Slow query highlighting** section, you can enter a baseline millisecond integer that the Devel module will use as its benchmark. Devel will highlight any query that takes longer than your sourced and benchmarked millisecond specification. This can also help you decide what needs caching on the site. Right now, we'll leave this set to the default of **5** millisecond.
5. You can also choose to store statistics about queries in your Devel database. We'll enable this and store every page view to start, as we're working on a development site. On a production site, you may not want to capture every page view, for example, raising this to capture every twentieth page view. This will help to stop our database table from overloading with statistics, but still allow us to capture some.

Query log

Collect query info
Collect query info. If disabled, no query log functionality will work.

Display query log
Display a log of the database queries needed to generate the current page, and the execution time for each. Also view are summed in the # column, and printed in red since they are candidates for caching.

Sort query log:
 by source
 by duration
The query table can be sorted in the order that the queries were executed or by descending duration.

Slow query highlighting:

Enter an integer in milliseconds. Any query which takes longer than this many milliseconds will be highlighted in the query, or a candidate for caching.

Store executed queries
Store statistics about executed queries. See the devel_x tables.

Sampling interval:

If storing query statistics, only store every nth page view. 1 means every page view, 2 every second, and so on.

6. The next setting is the URL for your API documentation. We'll just leave this set to the default Drupal API URL, which is `api.drupal.org`. Some developers store all of their API documentation locally, so if you do this you would want to enter your local URL/path here.
7. By checking the **Display page timer** box we can see page execution times in our query logs.
8. Check the box next to **Display memory usage**. We'll want to see how much memory we're using to generate pages.
9. Check the **Display redirection page**. This will prevent any query logs from being lost if a `drupal_goto()` function is run. We'll be presented with a redirect hyperlink path to the page that Drupal wants to direct us to after it shows us the query information.
10. Check the **Display form element keys and weights** box.
11. Another nice option here is the ability to skin your debug information. The **Krumo display** radio buttons allow you to pick a skin for your interface. Go ahead and choose one. I'll go with orange.

12. Because we're in the course of developing this site, let's check the box **Rebuild the theme registry on every page load**. We would not want to do this on a production site, as it could cause significant sluggishness, but while we're in development it's okay.

API Site:

The base URL for your developer documentation links. You might change this if you run `api.module` locally.

Display page timer
Display page execution time in the query log box.

Display memory usage
Display how much memory is used to generate the current page. This will show memory usage when `devel_init()` is been compiled with the `--enable-memory-limit` configuration option for this feature to work.

Display redirection page
When a module executes `drupal_goto()`, the query log and other developer information is lost. Enabling this setting the log can be examined before continuing to the destination page.

Display form element keys and weights
Form element names are needed for performing theming or altering a form. Their weights determine the position keys and weights beside each form item.

Krumo display:

default
 blue
 green
 orange
 schablon.com
 disabled

Select a skin for your debug messages or select *disabled* to display object and array output in standard PHP format.

Rebuild the theme registry on every page load
While creating new templates and theme_ overrides the theme registry needs to be rebuilt.

13. Expand the Administration menu settings. This allows you to tell the Devel module to **display additional information next to each menu item** on your site. For now let's leave this unchecked.
14. **Devel node access debug mode** allows you to have Devel module test node access permissions. As it states here, this can cause significant performance overhead. So for now, even on our development site, let's leave this unchecked. If we want to check the node access debug, we can come back and enable it later.
15. Select the default **SMTP library**.

16. Use the **Standard drupal error handler**. Backtrace will give you a nicer display of your debug information, but you'll need to install the Krumo library in order to use this.

The screenshot shows the 'Administration menu settings' section of the Devel module configuration. It includes the following settings:

- Display additional data for each menu item:**
 - None
 - Menu link ID
 - Weight
 - Parent link ID

Display the selected items next to each menu item link.
- Display all menu items** (checkbox): If enabled, all menu items are displayed regardless of your site permissions. Note: Do not enable on a production site.
- Devel Node Access debug mode** (checkbox): Debug mode verifies the grants in the node_access table against those that would be set by running [Rebuild permissions]. Considerable overhead.
- SMTP library:**
 - Default
 - Log only
- Error handler:**
 - None
 - Standard drupal
 - Backtrace

Choose an error handler for your site. Backtrace prints nice debug information when an error is noticed, and you choose krumo library. None is a good option when stepping through the site in your debugger.

17. Save configuration.

Inspecting database queries and Devel results

Immediately after you save your Devel settings configuration, you should see the results. The Devel settings page will reload and you will see a message concerning the user being redirected to that page at its URL here. The user is being redirected to <http://variantcube.com/fire/admin/settings/devel>. This shows you that the setting to show any redirections is working. This redirection did work. You'll also see a log entry on the page that shows the **Views plugins build time**. This would be more meaningful on a Drupal Views page that is using a Views plugin. The message we see here is **Views plugins build time: 27.6119709015 ms**

The meat and potatoes of our settings are visible at the very bottom of our Drupal page. If you scroll down, you'll notice all of the database query information. All of the query logged information presents itself in a styled table directly under your content. This can be quite intimidating the first time you view the table, as there is a massive amount of information presented. Remember that Drupal is a content management framework that is completely database driven. So when you view the Devel logs, you are really viewing all of the database work that Drupal is doing. This is excellent evidence of all the database queries being run and all the work Drupal is doing while integrating with MySQL.

The table starts out with a header that tells you how many queries were run on this specific page load and how long the entire query run took in milliseconds. Here we see that 74 queries ran in 362.98 milliseconds. Then, it tells us that our longer queries, those that are longer than 5 milliseconds, will be highlighted in red in the table report. These are the queries we want to pay closer attention to, as they take longer to run.

Views plugins build time: 29.2251110077 ms			
Executed 74 queries in 362.98 milliseconds. Queries taking longer than 5 ms and queries executed more than once, are highlighted . Page execution time was 2154.23 ms.			
ms # where query			
0.17	1	module_list	SELECT name, filename, throttle FROM system WHERE type = 'module' AND status = 1 AND bootstrap = 1 ORDER BY weight ASC, file
0.2	1	drupal_get_filename	SELECT filename FROM system WHERE name = 'user' AND type = 'module'
0.16	1	drupal_lookup_path	SELECT COUNT(pid) FROM url_alias
0.1	1	drupal_lookup_path	SELECT src FROM url_alias WHERE dst = 'admin/settings/devel' AND language IN('en', '') ORDER BY language DESC
0.21	1	module_list	SELECT name, filename, throttle FROM system WHERE type = 'module' AND status = 1 ORDER BY weight ASC, file
0.24	1	drupal_lookup_path	SELECT dst FROM url_alias WHERE src = 'devel/source' AND language IN('en', '') ORDER BY language DESC
0.57	1	cache_clear_all	DELETE FROM cache WHERE cid LIKE 'theme_registry%'
0.28	1	flood_is_allowed	SELECT COUNT(*) FROM flood WHERE event = 'devel_rebuild_registry_warning' AND hostname = '76.106.37.121' AND timestamp <= 1408300000
0.1	1	menu_get_item	SELECT * FROM menu_router WHERE path IN ('admin/settings/devel', 'admin/settings/%', 'admin/%/devel', 'admin/%/%', 'admin') ORDER BY fit DESC LIMIT 0, 1
0.17	1	drupal_lookup_path	SELECT dst FROM url_alias WHERE src = 'admin/settings/error-reporting' AND language IN('en', '') ORDER BY lan
■ Themer info	drupal_lookup_path		SELECT dst FROM url_alias WHERE src = 'admin/content/node-settings' AND language IN('en', '') ORDER BY lan

You can also inspect the table for repeated and duplicated queries, as all the queries are listed under the query column, and the table also tells you where the query took place (you can search for bottlenecks and repeated queries in the table).

One example we can point out here is the theme registry cache clear that Drupal runs based on the Devel module setting we chose. We told Devel to clear the theme registry on each page load. When the settings page reloaded after we made our settings choices, the query shows in the table. Look for the `cache_clear_all` function in the **where** column. This query took **.5 ms** to run and it cleared out the theme registry. The query reads: `DELETE FROM cache WHERE cid LIKE 'theme_registry%'`

If you have any doubts about the functions in the **where** column, you can click on the function link and it will take you to the API documentation that we enabled when we saved our Devel module settings. When I click on that link, the API function document loads at `api.drupal.org` here: http://api.drupal.org/api/function/cache_clear_all/6. The CID is set to `theme_registry` here, so that is what Drupal clears. The documentation tells us this, explaining the parameter in detail: **\$cid If set, the cache ID to delete. Otherwise, all cache entries that can expire are deleted.**

Scroll down in the query table to see if you can locate any potential issues or bottlenecks, especially looking for any red highlighted millisecond columns. Looking through the table, there appear to be two longer queries run for `cache_set`. Both of these come in at over 5 ms. This is probably not severe because this is the caching that we've set in our Drupal core cache settings. We may want to keep an eye on this as we view other page loads. You could also disable caching on the site to see if this makes a difference in the millisecond time.

0.97	1	cache_get	SELECT da
6.41	1	cache_set	UPDATE ca form\";s: path\";s: \\"templat \";}s:15: \\"type\"; functions \\"devel_t \";N;}s:8: \\"theme p \\"templat {s:9:\\"ar \\"devel_t:
6.01	1	cache_set	INSERT IN {s:24:\\"b \block.ad

The module also reports on queries which are executed more than once on your page load. These will also be highlighted in red. For example, if I go to the main Administer page of my site and load the page, then the query table reports that `_update_cache_get` query ran twice. The query did not take over 5 milliseconds, but it did get duplicated. The # column shows this with a red 2.

Time	#	Query	SQL Statement
0.56	1	<code>_update_cache_get</code>	<code>SELECT data, created, expire, serialized</code>
0.57	2	<code>_update_cache_get</code>	<code>SELECT data, created, expire, serialized</code>
0.98	2	<code>_update_cache_get</code>	<code>SELECT data, created, expire, serialized</code>
0.21	1	<code>drupal_lookup_path</code>	<code>SELECT dst FROM url_alias WHERE src = 'a'</code>

At the bottom of the page, under the entire table of database queries and logs, the summary is shown. It should look something like this:

Memory usage:

Memory used at: `devel_init()=1.77 MB, devel_shutdown()=32.58 MB.`

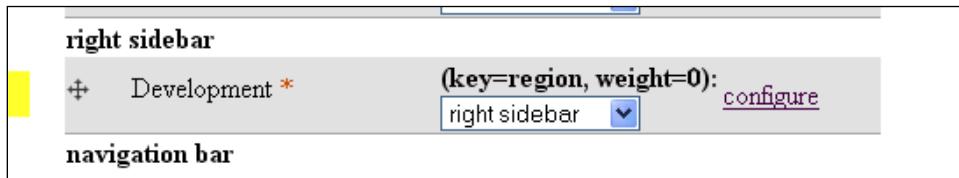
This reports the total memory used to load this page and all of its queries (**1.77 MB**). Another nice feature of this overall memory usage report is that you can run Devel on your pages and check to make sure your memory usage is consistent. So, if all of your pages are using a consistent **1.77 MB** approximate of memory, then you know that all pages are equally optimized and that your performance is also optimized. If you notice one page consuming more memory or large amounts of memory, this is a potential bottleneck that you will want to investigate in more detail.

Enabling the Devel module block

When you install the Devel module, the module will add a block to your site that you can enable and use as a site admin. The block allows you to get to Devel module functionality quickly. To enable the block follow these steps:

1. Go to **Site building | Blocks** here: <http://variantcube.com/fire/admin/build/block>
2. In your disabled blocks region look for the **Development** block and configure it. On the configuration screen you can check the block for an authenticated user or a site admin role (if you've created one) under the **Show block for specific roles**. This will prevent any anonymous user or default Drupal authenticated user from viewing the Devel block.

3. Save your block configuration.
4. To enable your block, click in the region drop-down box and select **right sidebar**, so the block will appear in the right sidebar of your site.



5. Save blocks.

Your Development block will now be visible in your right sidebar. I'm currently viewing the Devel block in my Apollo Zen sub-theme. You should see something like this:

A screenshot of the Development block in the right sidebar. The title 'Development' is at the top. Below it is a list of links:

- ▶ [Database queries](#)
- [Disable Theme developer](#)
- [Empty cache](#)
- [Execute PHP Code](#)
- [Function reference](#)
- [Hook elements\(\)](#)
- [Node access summary](#)
- [PHPinfo\(\)](#)
- [Rebuild menus](#)
- [Reinstall modules](#)
- [Run cron](#)
- [Session viewer](#)
- [Theme registry](#)
- [Variable editor](#)

You may have noticed that the Development block does not show if you navigate to an admin section of your site and have the Garland theme installed as the default administration theme. The workaround for this is to disable your custom theme, enable Garland as your default theme, and then re-enable the Development block in Garland in the right sidebar. If you are working on a production site, you may want to take your site offline before making this tweak.

Using the Devel module block

The Devel block gives you quick access through links to your Devel functionality. You can perform the following with the block:

- Access your database queries on one page report view
- Disable your Theme developer and re-enable it
- Empty your entire site performance cache
- Execute PHP code
- Access a function reference
- Access `Hook_elements()`
- Access your `node_access` summary
- Get information on your PHP settings
- Rebuild your site menus
- Reinstall your modules
- Run your cron
- View sessions
- Clear your Theme registry
- Edit variables

Let's look at each of these tools briefly. The best way to learn how to use the Devel block is to start accessing all of its functionality through the block links and to try out each component.

Database queries

We've already discussed Database queries in detail, but if you click on this link in the Devel block, it will launch a page of all the database queries that the Devel module is running. This is another way to get to those queries.

Empty cache

Clicking on the **Empty cache** link will clear your site's performance cache. This is a nice, quick, and easy method of clearing your cache. Once run, the **Empty cache** page will load with a message that your **Cache cleared**.

The screenshot shows a web page titled "Empty cache". At the top, there is a blue header bar with the word "Home". Below the header, the main content area has a light blue background. The title "Empty cache" is displayed in bold black font. Underneath the title, a green-bordered box contains the text "Cache cleared.". Below this box, a message states "The user is being redirected to <http://variantcube.com/fire/devel/php>". At the bottom of the page, there is some small text that reads "Views plugins build time: 2.79998779297 ms".

Disable/Enable Theme developer

If you want to disable your theme developer functionality, click on the **Disable Theme developer** link in the Development block. When you do this, you'll notice that your theme developer box will disappear. To enable it again, click on the **Enable Theme developer** link.

Execute PHP code

To execute PHP code, click on the link and you'll be presented with a page that has a text area box where you can paste your PHP code. For example, go ahead and execute a simple PHP info script. Execute this code, but remember not to add any PHP tags. You do not need to surround your code with opening and closing PHP brackets. Just type or paste in:

```
phpinfo();
```

Click on the **Execute** button. Your code will execute. The results will not look as pretty as the PHP info page you get through your Status report. Basically, here Drupal will output the results in HTML without any style formatting rendered. So, you'll see your PHP info report, but it will be surrounded by HTML code. Here's what I see after executing the script:

```
<tr><td class="e">auto_prepend_file</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">browscap</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">default_charset</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">default_mimetype</td><td class="v">text/html</td><td class="w">text/html</td></tr>
<tr><td class="e">define_syslog_variables</td><td class="v">0ff</td><td class="w">0ff</td></tr>
<tr><td class="e">disable_classes</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">disable_functions</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">display_errors</td><td class="v">On</td><td class="w">On</td></tr>
<tr><td class="e">display_startup_errors</td><td class="v">0ff</td><td class="w">0ff</td></tr>
<tr><td class="e">doc_root</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">docref_ext</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">docref_root</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">enable_dl</td><td class="v">On</td><td class="w">On</td></tr>
<tr><td class="e">error_append_string</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">error_log</td><td class="v">error_log</td><td class="w">error_log</td></tr>
<tr><td class="e">error_prepend_string</td><td class="v"><i>no value</i></td><td class="w"><i>no value</i></td></tr>
<tr><td class="e">error_reporting</td><td class="v">6135</td><td class="w">6135</td></tr>
<tr><td class="e">expose_php</td><td class="v">On</td><td class="w">On</td></tr>
<tr><td class="e">extension_dir</td><td class="v">/usr/local/lib/php/extensions/no-debug-zts-20060613</td><td class="w"></td></tr>
<tr><td class="e">file_uploads</td><td class="v">On</td><td class="w">On</td></tr>
<tr><td class="e">highlight.bg</td><td class="v"><font style="color: #FFFFFF">#FFFFFF</font></td><td class="w"></td></tr>
<tr><td class="e">highlight.comment</td><td class="v"><font style="color: #FF8000">#FF8000</font></td><td class="w"></td></tr>
<tr><td class="e">highlight.default</td><td class="v"><font style="color: #0000BB">#0000BB</font></td><td class="w"></td></tr>
<tr><td class="e">highlight.html</td><td class="v"><font style="color: #000000">#000000</font></td><td class="w"></td></tr>
<tr><td class="e">highlight.keyword</td><td class="v"><font style="color: #007700">#007700</font></td><td class="w"></td></tr>
```

Function reference

The **Function reference** link will load a page that shows a list of all the user functions that are generated during a page load on the site. It's a long list on our site covering all functions from `_advanced_help_parse_ini` to `xmlrpc`. You can click on a function link to view more details about it and read its documentation. For example, if you click on the `_comment_form_submit` function link, you'll load the function page at `api.drupal.org` which is here: http://api.drupal.org/api/function/comment_form_submit/6. This will explain the function and give you an example of its use.

Function reference

The theme registry is being rebuilt on every request. Remember to turn off this feature on production websites.

This is a list of defined user functions that generated this current request lifecycle. Click on a function name to view its documentation.

- [_advanced_help_parse_ini](#)
- [_block_get_cache_id](#)
- [_block_rehash](#)
- [_block_themes_access](#)
- [_blog_post_exists](#)
- [_book_add_form_elements](#)
- [_book_flatten_menu](#)
- [_book_link_defaults](#)
- [_book_outline_access](#)
- [_book_outline_remove_access](#)
- [_book_parent_depth_limit](#)

Hook_elements()

`Hook_elements` launches a page that displays all of your hook arrays. This will be presented in a nice Krumo-based screen using the orange skin we selected when we configured our Devel module.

You can click on an array in the table to view its corresponding elements.

Hook_elements()
... (Array, 36 elements)
form (Array, 2 elements)
#method (String, 4 characters) post
#action (String, 20 characters) fire/devel/elements
submit (Array, 5 elements)
#input (Boolean) TRUE
#name (String, 2 characters) op
#button_type (String, 6 characters) submit
#executes_submit_callback (Boolean) TRUE
#process (Array, 1 element)
button (Array, 5 elements)
image_button (Array, 7 elements)
textfield (Array, 5 elements)
password (Array, 4 elements)
password_confirm (Array, 2 elements)

PHPinfo()

`PHPinfo()` launches your site's default PHP information page. This is the same PHP information page that you can launch from your site's Status report. The Devel block gives you easier access to it, so you do not have to load your Status report page each time you need information on your PHP settings.

Rebuild menus

Rebuild menus flush your menu configuration, return your menu items to their default status, and remove any customization you've made to a specific menu item. Be careful with this setting, as you will most likely not want to do this on a production site. However, this may prove helpful on a development site if you need to troubleshoot why menus are not behaving correctly.

Reinstall modules

This is another Development block tool that you need to be careful with. It provides easy access to your modules admin, and you can select any of your core or contributed modules and reinstall them here. But beware that if you do this, the module reinstallation will wipe out and override any customizations you had made. Reinstalling modules will also wipe out any data you have in your module's database tables, so be sure to only use this functionality if you know what you are doing as an advanced module developer. This is a good tool for module developers.

Running cron

This link provides easy access to running cron manually on your site without having to load your Status report.

Session viewer

This shows you the contents of your session variable. Drupal session variables are variables that store site visitor information in the Drupal application framework for the entire length of the logged in user's session. So, as soon as you logout of the site, your session variable will expire. The variable may also expire after a certain period of inactivity on the website. These variables help to store information about a specific authenticated site visitor who is logged into your site.

So, when I'm logged into this site at this moment as the admin user, the session name and session ID is shown. The Session name is **SESS701436a5b5363d46c4148943e8686c89**, and the Session ID is **8582e26c836f16d48b0c1048bfd**. Notice here that if I logout of the site and then login, and run the **session viewer** link again I will generate different name and ID numbers.

The screenshot shows a web page titled "Session viewer". The page displays the contents of the \$_SESSION variable, which is currently an empty array. Below this, it shows the session name and session ID. The session name is "SESS701436a5b5363d46c4148943e8686c89" and the session ID is "f663e230f35b1a69e10f26c1cd61c19". The page is identified as being from "Krumo version 0.2a | http://krumo.sourceforge.net".

Session name	Session ID
SESS701436a5b5363d46c4148943e8686c89	f663e230f35b1a69e10f26c1cd61c19

Theme registry

Theme registry shows you all of your theme arrays and corresponding elements. This is helpful for the theme developer. For example, you can get theme function names here by clicking to expand an array and looking at its elements and functions. If we click on **Image**, we can see that the function, **theme_image**, is listed.

Variable editor

Finally, the last link in the Development block is the Variable editor. This shows you all of the site variables that are stored in the database. This is a very helpful list. I can glance quickly through the alphabetical list looking for the `site_name` variable name and locate its value. The value is the name of our site, for example, **Fire Trucks of Maryland and Pennsylvania**. So, this table gives you the variable name along with its value. Another example is the administration theme we're using. The variable is called `admin_theme` and the value for this is `s: 7 : "garland" ;`. This shows that the Garland theme is our current administration theme.

The Variable editor tool also provides you with an Edit link that allows you to edit the variable on the fly, using the Devel module. You can also select the checkbox next to the variable name to delete it, but beware that if you delete the variable here it will delete it from the database.

<input type="checkbox"/>	site_mail	s:22:"trevor@variantcube.com";	30	edit
<input type="checkbox"/>	site_mission	s:0:"";	7	edit
<input type="checkbox"/>	site_name	s:40:"Fire Trucks of Maryland and Pennsylvania";	48	edit
<input type="checkbox"/>	site_offline	s:1:"0";	8	edit
<input type="checkbox"/>	site_offline_message	s:105:"Fire Truck Photos is currently under maintenance. We shoul...";	114	edit
<input type="checkbox"/>	site_slogan	s:0:"";	7	edit

Click on the **Edit** link next to the `site_name` variable. This will launch a page with a text area box. The box will contain the current value of the variable. Go ahead and delete the value and add a new one, or edit the one that is there. Then click on **Submit**. This is a wonderful tool that allows you to quickly locate site variables and edit their values without having to search through the Drupal administration for the specific node, block, or menu, or other configuration that contains the variable. You can get the entire list here and edit them. I'm going to tweak the value to be **Fire Trucks of Maryland and Pennsylvania**. Then I'll click on **Submit** and refresh my site page, and the new site name should be visible.

site_name

Old value (key=value, weight=0):

Fire Truck Photos

New value (key=new, weight=0):

Fire Trucks of Maryland and Pennsylvania

≡≡≡

Here's a screenshot of the result:



Summary

We've covered a lot of material in this chapter and looked at one of the larger Drupal contributed modules—a module that provides a huge amount of functionality and tools to the Drupal performance developer. You should now take time to use the Devel module every day during your Drupal development work and get to know each component in greater detail. Here's what we covered:

- We looked at recent log entries in order to check on any potential PHP errors or other parse errors that may be occurring on our site. We also looked at the top page not found and access denied errors.
- We installed and configured the Devel module and set permissions for our admin users to use the module.
- We used the Devel module to check on database queries that occur during our page loads and how long these queries take to run. We also looked for duplicate queries and learned to watch out for those queries that take longer than our threshold millisecond benchmark.
- We used the Development block to get easier access to some common performance tools such as the manual Drupal cron task, clearing our performance cache, checking our theme registry, and tweaking our site variables.

Let's take a break! When we come back in Chapter 4, we'll discuss how best to implement the trifecta of Drupal modules CCK, Views, and Panels for best performance on our site. We'll also learn how to clear our Views and Panels caches, and to throttle content, blocks, and other elements on our site.

4

Performance Optimization

In this chapter, we're going to learn how to throttle modules and blocks on our site in order to increase performance and reduce server load during high traffic periods. We will use the **Throttle module** to do this.

We'll return to a discussion of the **Devel module** and **how it can be used to generate dummy content, users, and taxonomy categories for our development site**. Using Devel to generate test or dummy content provides an easy method of building a test or demo site quickly.

We'll look at caching mechanisms in two of the larger scale contributed modules, Views and Panels, which you may find yourself using frequently. **Views** and **Panels** both allow you to cache the data and content that you insert into these modules. We'll look at how these caching mechanisms work and how you can maintain your cache in both modules.

To summarize, here's what we'll be doing in this chapter:

- Throttling modules and blocks
- Using the Development module to generate taxonomy, users, and content
- How to run Views 2.x for best performance, including how to clear your Views cache
- Panels module caching mechanisms and how to maintain your Panels cache.

Enabling and configuring the Throttle module

Drupal allows you to control when your modules and blocks get enabled and shown to your site visitors. This helps you to prevent bottlenecks in your server's web traffic and to optimize your server load to prevent any congestion that it might experience with its bandwidth and traffic. Throttling blocks and modules becomes increasingly important on larger scale websites where you have many blocks and modules active. You may have a site that contains a large number of blocks, for example, that have been built with the Views module. You can throttle these blocks, so they only get enabled when the site visitor calls a page that is supposed to show that block. The throttle module allows you to configure it, so it automatically gets enabled when the usage of your site goes above a certain threshold. For example, this can be the number of anonymous users visiting your site. When a certain amount of visitors are on your site, you can have Drupal enable throttling.

Using the Throttle module is essential on shared servers where you may not have all of the resources on the server made available to you at any given time or on a server that gives you limited CPU resources and bandwidth. You may not need to use Throttle on higher performance-dedicated servers because they will most likely be providing you with good performance. But on shared servers it does become important to use Throttle.

If you did not enable the Throttle module after we upgraded our site to Drupal 6.13, we need to enable it first. Once enabled, we can then configure the module. Follow these steps:

Under your Core-optional module list, check the box next to **Throttle** and then save your module configuration.



Throttle

6.13

Handles the auto-throttling mechanism, to control site congestion.

There are two methods of accessing the Throttle module configuration. You can visit the main Throttle configuration page to set auto throttling settings for your site. Also, you can enable throttling for each module and block on your site. We'll look at both methods now. Note that your modules admin page explains how to access and enable both types of throttling (module and auto) at the top of the page in its introductory help text. You will only see your module throttle checkboxes available if you have enabled the Throttle module first.

Configuring the Throttle module for auto throttling features

Go to **Site configuration | Throttle** to load your Throttle module settings form or click on the **throttle configuration page** link through your main modules admin page.

The Throttle configuration page explains what the Throttle module does and gives you a link to more information through the **more help** link. If you click on that link and have the Advanced Help module active, you will launch a detailed Throttle module help and explanation page.

On this page you can configure three throttle elements that fall under the default Throttle module congestion control feature:

- Auto-throttle on anonymous users
- Auto-throttle on authenticated users
- Auto-throttle probability limiter

Auto-throttle on anonymous users allows you to set a threshold for enabling your congestion control throttle dependent on anonymous user activity. So, for example, you will want to choose a threshold number of anonymous users to enter into this field. When this number of anonymous users is reached, the auto-throttle feature will be enabled. If you want the auto-throttle to be enabled after 250 anonymous users are on your site at the same time, you can type **250** into this field. Set this field to **0** if you do not want to use the auto-throttle feature.

Drupal also tells you here that you can determine how many users are on your site at any given time by enabling the **Who's Online** block – this will show you all of the anonymous users who are browsing your site and authenticated users who are logged into your site.

The **Auto-throttle on authenticated users** works using the same method. Add the threshold number of authenticated users that you want logged into your site before the point your throttle gets enabled. We'll set this to **50** authenticated users.

Throttle

The throttle module provides a congestion control mechanism that automatically adjusts to a surge in incoming traffic. If your site is referenced by a popular website, or experiences a "Denial of Service" (DoS) attack, your webserver might become overwhelmed. The throttle mechanism is utilized by modules to temporarily disable CPU-intensive functionality, increasing performance.

[\[more help...\]](#)

Auto-throttle on anonymous users (`key=throttle_anonymous, weight=0`):

The congestion control throttle can be automatically enabled when the number of anonymous users currently visiting your site exceeds the specified threshold. For example, to start the throttle when your site has 250 anonymous users online at once, enter '250' in this field. Leave this value blank or set to "0" if you do not wish to auto-throttle on anonymous users. You can inspect the current number of anonymous users using the "Who's online" block.

Auto-throttle on authenticated users (`key=throttle_user, weight=0`):

The congestion control throttle can be automatically enabled when the number of authenticated users currently visiting your site exceeds the specified threshold. For example, to start the throttle when your site has 50 registered users online at once, enter '50' in this field. Leave this value blank or set to "0" if you do not wish to auto-throttle on authenticated users. You can inspect the current number of authenticated users using the "Who's online" block.

Auto-throttle probability limiter (`key=throttle_probability_limiter, weight=0`):

The auto-throttle probability limiter is an efficiency mechanism to statistically reduce the overhead of the auto-throttle. The limiter is expressed as a percentage of page views, so for example if set to the default of 10% we only perform the extra database queries to update the throttle status 1 out of every 10 page views. The busier your site, the lower you should set the limiter value.

The **Auto-throttle probability limiter** helps to reduce the overhead of the auto-throttle module. It's a built-in performance check just for this Throttle module. You can set a percentage of page views for your site. For example, if you set the page view percentage to **10%**, then the module will only perform extra database queries to update the Throttle module status once for every 10 page views. Drupal tells you that the busier your website, the lower you want to set this value. Leave it set to the default of **10%**. Save your auto-throttle configuration.

Throttling your modules

You can also throttle each of your core and contributed modules as long as they have a Throttle checkbox next to their line item on the modules admin page. Load your modules admin page and look for the Throttle checkboxes. This allows you to tell Drupal to throttle a specific module during high traffic periods on your site. This means that when your site reaches a high traffic threshold (based on the auto throttling settings you determined above) your site will temporarily disable the module in question. This will throttle the module until your site returns to a stable status.

You do need to be careful here. You should throttle those modules that are of lesser importance when your site reaches its threshold of user activity. When you throttle, you are temporarily disabling the module, so it will also temporarily disable that module's functionality during high server loads. So, you may want to disable some modules, such as Views, but leave your CCK module enabled so that your users can still see the content that is being filtered into the View. We'll go ahead and throttle the following modules:

- Administration menu (because this module is only being used by our logged in admins)
- Chaos tool suite (all submodules here)
- Comment
- Contact
- Database logging
- Help
- PHP filter
- Search
- Statistics
- Advanced Help
- FCKEditor
- IMCE
- Lightbox2
- Poormanscron

You can select more modules to throttle based on your preferences and the usage of your site. Use the above as an example and model to follow. Check the throttle boxes and save your configuration. During the next high server load period, these modules will be disabled temporarily to increase the performance of your site during its high server load period.

The screenshot shows the 'Modules' administration page in Mozilla Firefox. The URL is <http://variancube.com/fire/admin/build/modules>. The page lists various modules with columns for Enabled, Throttle, Name, Version, and Description. An arrow points to the 'Throttle' checkbox for the 'Views content panes' module in the 'Chaos tool suite' section.

Enabled	Throttle	Name	Version	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Bulk Export	6.x-1.0-rc1	Performs bulk exporting of data objects known about by Chaos tools. Depends on: Chaos tools (enabled)
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Chaos tools	6.x-1.0-rc1	A library of helpful tools by Merlin of Chaos. Required by: Bulk Export (disabled), Page manager (enabled), Panels (enabled), Views content panes (enabled), Mini panels (enabled), Panel nodes (enabled), Bonus: Panels (enabled)
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Page manager	6.x-1.0-rc1	Provides a UI and API to manage pages within the site. Depends on: Chaos tools (enabled)
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Views content panes	6.x-1.0-rc1	Allows Views content to be used in Panels, Dashboard and other modules which use the CTools Content API. Depends on: Chaos tools (enabled), Views (enabled)

Enabled	Throttle	Name	Version	Description
<input type="checkbox"/>	<input type="checkbox"/>	Aggregator	6.13	Aggregates syndicated content (RSS, RDF, and Atom feeds).
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Blog	6.13	Enables keeping easily and regularly updated user web pages or blogs.
<input type="checkbox"/>	<input type="checkbox"/>	Blog API	6.13	Allows users to post content using applications that support XML-RPC blog APIs.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Book	6.13	Allows users to structure site pages in a hierarchy or outline.
<input type="checkbox"/>	<input type="checkbox"/>	Color	6.13	Allows the user to change the color scheme of certain themes.
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Comment	6.13	Allows users to comment on and discuss published content. Required by: Forum (disabled), Tracker (disabled)
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Contact	6.13	Enables the use of both personal and site-wide contact forms.
<input type="checkbox"/>	<input type="checkbox"/>	Content translation	6.13	Allows content to be translated into different languages. Depends on: Locale (disabled)
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Database logging	6.13	Logs and records system events to the database.

Throttling blocks

You can also throttle your blocks. To do this, go to the Blocks admin page here: [Administer | Site building | Blocks](#). You'll notice that there is a new checkbox selection for Throttle. You can choose which blocks to throttle by checking the **Throttle** checkbox next to each of your enabled blocks. We'll go ahead and throttle all of our blocks except for the **User login**, as we still want to allow users to login to the site during high traffic periods. The throttle functionality works the same here as it does with modules. These blocks will be temporarily disabled during high site traffic.

Block	Region	Throttle	Operations
Left sidebar			
⊕ Navigation	(key=region, weight=0): Left sidebar ▾	<input checked="" type="checkbox"/> (key=throttle, weight=0)	configure
⊕ User login	(key=region, weight=0): Left sidebar ▾	<input type="checkbox"/> (key=throttle, weight=0)	configure
Right sidebar			
⊕ Development	(key=region, weight=0): Right sidebar ▾	<input checked="" type="checkbox"/> (key=throttle, weight=0)	configure
Content			
No blocks in this region			
Header			
No blocks in this region			
Footer			
⊕ Devel Node Access	(key=region, weight=0): Footer ▾	<input checked="" type="checkbox"/> (key=throttle, weight=0)	configure

Once you check your throttle boxes, save your blocks configuration. The next time you have high site traffic, these blocks will be temporarily disabled.

Generating test users, categories, and content

Another use of the Devel module besides all the great functionality we discussed in Chapter 3 is to generate test 'dummy' content, taxonomy categories, and users for your website. This functionality is part of the Devel module and to confirm that you can use this, go to your admin modules list and check to make sure the Devel generate module is enabled.

To generate users, nodes, and taxonomy terms for your site go to the **Generate items** main page in your site admin. That will launch a page with links to **Generate categories**, **Generate content**, and **Generate users**. As we already have some content on our site (that I originally implemented by using **Generate content**), we're going to try generating some test users for our website. Follow these steps to generate users:

1. Click on the **Generate users** link.
2. Tell the Devel module how many users to add to the site. The default is **50**. We'll add 10 users to our site. Type **10** in the field.
3. You can also tell Drupal to kill or delete all the other users (if you've already added test users) on your site apart from the super user admin (with User ID 1). This is helpful. You can add some test users and then easily delete them using this module. As we do not have any users on our site besides the super user, we'll leave this box unchecked.
4. The Devel module will randomly generate a user account history time for each account. You can tell Devel how far back to go in the select box under the **How old should user accounts be?**. We'll change this to **4 weeks ago**.
5. Click on the **Do it!** button.

The screenshot shows a web form titled "Generate users". It has two main sections: "How many users would you like to generate? (key=num, weight=0)" with a text input field containing "10", and "How old should user accounts be? (key=time_range, weight=0)" with a dropdown menu set to "4 weeks ago". Below the dropdown is a note: "User ages will be distributed randomly from the current time, back to the selected time." At the bottom is a blue "Do it!" button.

As soon as you click on the **Do it!** button, the user accounts will be created automatically by the Devel module.

The screenshot shows a confirmation message: "10 users created." followed by a redirect notice: "The user is being redirected to <http://variantcube.com/fire/admin/generate/user>".

Go to your Users list to see the dummy users that the Devel module created for you. They should all have usernames and be set to active status.

You can also use the **Generate items** functionality of the Devel module to generate content and categories. Click on each of these links to try out this functionality. When you click on **Generate categories**, the module will ask you about **vocabularies**, **terms**, and the **max word length** for a term or **vocab** name. Plug some numbers into these fields and then click on the **Do it!** button. Let's generate **3 vocab**s and **10 terms per vocab**. The **Max word length of term/vocab names** will be set to the default of **12** characters.

Generate categories

How many vocabularies would you like to generate? (key=num_vocab, weight=0):
3

How many terms would you like to generate? (key=num_terms, weight=0):
10

Max word length of term/vocab names (key=title_length, weight=0):
12

Delete existing terms and vocabularies before generating new content. (key=kill_taxonomy, weight=0)

Do it!

Drupal will tell you it created the new vocabs and terms. You can browse to your taxonomy page to view the vocabs and terms.

Generate categories

- Created the following new vocabularies:
 - led
 - uebugeswadr
 - lestamathasw
- Created the following new terms:
 - bewrusta
 - babodumen
 - lobrechap
 - clupruclosta
 - wre
 - cride
 - frostip
 - uaswe
 - drawrugephu
 - podushe

Once you click on **Do it!**, browse to your Taxonomy admin list and you should now see the new vocabularies and terms in the site that were generated using the Devel module.

Now that you have added some dummy taxonomy content using the Devel module, you can view your taxonomy vocabs and terms by visiting **Content Management | Taxonomy**. On this page you'll see the new list of vocabs. If you click on the **list terms** link, you'll see all of the tags per vocab that were added.

To generate test content go to the **Generate content** page, fill out the form fields for the types of node content you'd like to create, how many nodes, the maximum number of generated comments, whether you want to add taxonomy terms to each node, and URL aliases per node. This also allows you to delete any previously posted test content that you may have integrated with your site using the Devel module before. Fill out all the fields you need and then click on **Do it!**. The test content will be generated.

For our next example using the Views module (in the next section on *Views caching*), I'm going to generate Blog entry test nodes along with the corresponding taxonomy. Generate your own test nodes that you will use in the next section on the Views module. I'm going to go ahead and delete all of my previous test blog content so that I can start afresh with brand new content. I'll generate the default **50** nodes of content. Nodes will be dated as far back as 4 weeks ago. I'll generate a maximum number of **10** comments per blog post. I will add taxonomy terms, upload functionality, and URL aliases to each node. Click on **Do it!** button.

Generate content

Which node types do you want to create? (key=node_types, weight=0):

Blog entry
 Panel
 Book page
 Page
 Photo Gallery. This type contains CCK fields which will only be populated by fields that implement the content_generate hook.
 Story

Delete all content in these node types before generating new content. (key=kill_content, weight=0)

How many nodes would you like to generate? (key=num_nodes, weight=0):

Rebuild menu based on this return to their default set

How far back in time should the nodes be dated? (key=time_range, weight=0):

▼
 Node creation dates will be distributed randomly from the current time, back to the selected time.

Maximum number of generated comments per node (key=max_comments, weight=0):

You must also enable comments for the node types you are generating.

Max word length of titles (key=title_length, weight=0):

Add an upload to each node (key=add_upload, weight=0)
Requires upload.module

Add taxonomy terms to each node. (key=add_terms, weight=0)
Requires taxonomy.module

Add an url alias for each node. (key=add_alias, weight=0)
Requires path.module

Generate node view statistics (node_counter table). (key=add_statistics, weight=0)

Drupal will generate the content and you'll see a progress task bar as the content batch is generated. Then, Drupal will show you a confirmation screen showing a list of all the blog entry content that was created.

Generate content

- Blog entry *Capo Hendrerit Augue Vel Nunc* has been deleted.
- Blog entry *Si Interdico Lucidus Haero Quidne Nobis Aliquip* has been deleted.
- Blog entry *Virtus Venio Venio Rusticus Roto Quidne Et Rusticus* has been deleted.
- Blog entry *Consequat Gravis Abbas Aliquip Bene* has been deleted.
- Blog entry *Hendrerit Te Gemini* has been deleted.
- Blog entry *Wisi Pagus Modo Venio Tamen* has been deleted.
- Blog entry *Quidem Usitas Patria Verto Pala* has been deleted.
- Blog entry *Enim Usitas Vero Haero Ymo Zelus Et Sino* has been deleted.
- Blog entry *Usitas Patria* has been deleted.
- Blog entry *Sino Olim Jugis Aliquip Capto Premo Sit* has been deleted.

If you browse to your main content admin list, you will also see all the new blog entries that were created using Devel.

Now that we have test content, categories, and terms generated in our site, we're now ready to build a View using our test content and to start using View caching to increase the performance of our Views. We can also integrate our View with the Panels module and utilize some of the Panels module cache functionality.

Views caching

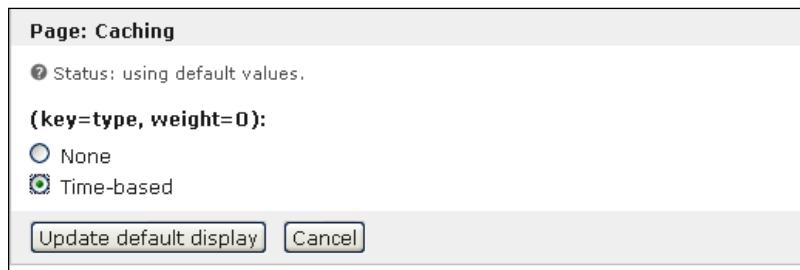
The Views 2 module allows you to cache your Views data and content. You can cache Views data per View. We're going to enable caching on one of our existing Views, and also create a brand new View and set caching for that as well using the test content we just generated. This will show you a nice integration of the Devel functionality with the Views module and then how caching works with Views.

Go to your **Site building | Views** configuration page and you'll see many of your default and custom views listed. We have a view on this site for our main photo gallery. The view is named **photo_gallery** in our View listing. Go ahead and click on one of your Views **edit** links to get into edit mode for a View.

In our Views 2 interface mode, we'll see our tabs for **default**, **Page**, and/or **Block** View display. I'm going to click on my **Page** tab to see my View's page settings. Under my **Basic settings** configuration, I'll see a link for **Caching**. Currently, our **Caching** link states **None**, meaning that no caching has been configured for this view.

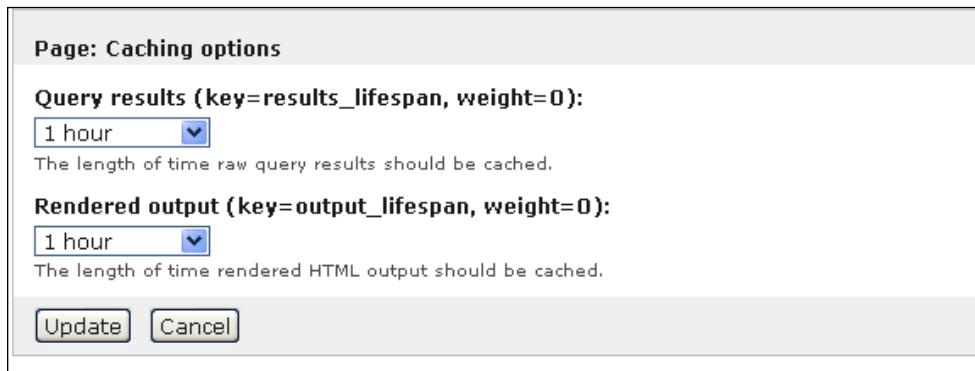


Click on the **None** link. Select the **Time-based** radio button. This will enable **Time-based** caching for our View page.



Click the **Update default display** button. The next caching options configuration screen will ask you to set the amount of time for both, your View **Query results** and for your View **Rendered output**. **Query results** refer to the amount of time raw queries should be cached. **Rendered output** is the amount of time the View HTML output should be cached. So basically, you can cache both your data and your frontend HTML output.

Set them both to the default of **1 hour**. You can also set one to a specific time and the other to **None**. Go ahead and tweak these settings to your own requirements. I'm leaving both set to the default of **1 hour**.



Click on the **Update** button to save your caching options settings.

You are now caching your View. Save your View by clicking on the **Save** button. The next time you look at your View interface you should see the caching time notation listed under your Basic settings. It will say **1 hour/1 hour** for this setting.

Once you enable Views caching, if you make a change to your View settings and configuration, the results and output of the View may not update while you have caching enabled. So, while in Views development you may want to disable caching and set it to **None**. Otherwise, this next section will show you how to disable your Views cache while you are in development.

To see the performance results of this, you can use the Devel module's functionality again. When you load your View after you enable caching, you should see a decrease in the amount of **ms** (milliseconds) needed to build your Views plugin, data, and handlers. So, if your Views plugin build loaded in **27.1 ms** before you enabled caching, you may notice that it changes to something less—for example, in my case it now shows that it loads in **2.8 ms**. You can immediately see a slight performance increase with your View build.

Antique Fire Apparatus & Muster, Harrisburg, PA

West Shore Engine

Tower 3

Tower 3

Tower 3

Tower 3

Engine 2

Engine 2

Engine 2

Views plugins build time: 2.8350353241 ms
Views data build time: 21.1479663849 ms
Views handlers build time: 1.93190574646 ms

←

Let's go ahead and build a brand new View using the test content that we generated with the Devel module and then enable caching for this View as well.

Go to your Views admin and follow these steps:

1. **Add** a new View.
2. Name the View, add a description and a tag if applicable.
3. Click on **Next**.
4. I'm going to create a View that filters my blog entries and lists the new blog entries in post date order using the Devel content I generated.

5. Add a **Page** display to your new View.
6. Name the page View.
7. Give the page View a **title**.
8. Give your View an **HTML list style**.
9. Set the View to display **5 posts** and to use a **full pager**.
10. Set your caching to **Time-based** (following instructions above in the first view we edited).
11. Give the view a **path**.
12. Add a **Node:Title** field and set the field to be linked to its node.
13. Add a filter in order to filter by **Node:Type** and then select **Blog entry**.
14. Set your **Sort criteria** to sort by **Node:Post date** in **ascending** order by hour.
15. Your settings should look similar to this:

The screenshot shows the 'Blog entries' view configuration in the Drupal admin interface. The left sidebar has a 'Page' display selected. The main area shows the 'Basic settings' tab with the following details:

- Name:** Blog entries
- Title:** Recent Blog entries
- Style:** HTML List
- Row style:** Fields
- Use AJAX:** No
- Use pager:** Yes
- Items per page:** 5
- Distinct:** No
- Access:** Unrestricted
- Caching:** 1 hour/1 hour
- Exposed form in block:** No
- Header:** None
- Footer:** None
- Empty text:** None
- Theme:** Information

Below this is the 'Page settings' tab with the path **Path: blog-recent** highlighted.

On the right side, there are four configuration sections:

- Relationships:** None defined
- Sort criteria:** Node: Post date asc
- Arguments:** None defined
- Filters:** Node: Type = Blog entry
- Fields:** Node: Title

At the bottom, there are 'Save' and 'Cancel' buttons, and a 'Live preview' section showing a list of blog entries with radio buttons next to them. The preview also includes a pager at the bottom.

Save your View by clicking on the **Save** button. Your new View will be visible at the **Page** path you gave it and it will also be caching the content and data it presents. Again, if you refresh your View page each time you should notice that the plugins, data, and handlers build times decrease or stay very similar and consistent in load times. You should also notice that the Devel database queries status is telling you that it's using the **cached results** and **cached output** for the View build times and the MySQL statements. You should see the following code sitting below your page content on the View page you are looking at. It will resemble this:

```
Views plugins build time: 23.509979248 ms
Views data build time: 55.7069778442 ms
Views handlers build time: 1.95503234863 ms

SELECT node.nid AS nid,
       node_data_field_photo_gallery_photo.field_photo_gallery_photo_fid
    AS node_data_field_photo_gallery_photo_field_photo_gallery_photo_fid,
       node_data_field_photo_gallery_photo.field_photo_gallery_photo_list
    AS node_data_field_photo_gallery_photo_field_photo_gallery_photo_list,
       node_data_field_photo_gallery_photo.field_photo_gallery_photo_data
    AS node_data_field_photo_gallery_photo_field_photo_gallery_photo_data,
       node.type AS node_type,
       node.vid AS node_vid,
       node.title AS node_title,
       node.created AS node_created
  FROM {node} node
 LEFT JOIN {content_type_photo} node_data_field_photo_gallery_photo ON
node.vid = node_data_field_photo_gallery_photo.vid
 WHERE (node.status <> 0) AND (node.type in ('%s'))
 ORDER BY node_created ASC
Used cached results
Used cached output
```

The next section will show you how to clear your Views cache while you are in development mode.

Clearing your Views 2 module cache

If you are working with the Views module in your site, you'll want to know how to clear your Views module cache so that updates and tweaks you make to your Views via the Views 2 UI show up immediately. The Views module will cache existing View configuration, settings, and content, so if you're not seeing the changes you've made or are experiencing other issues with your Views, you'll want to make sure to clear the cache.

Here are the steps to clear your Views cache:

1. Go to **Site building | Views** to access your Views admin.
2. At the top of the page you'll see a **Tools** button. Click on it.
3. The simplest method of clearing your Views cache is to click on the **Clear Views' cache** button.
4. You will receive a message: **The cache has been cleared.**

The cache has been cleared.

The user is being redirected to <http://variantcube.com/fire/admin/build/views/tools>.

Views plugins build time: 33.812046051 ms

There are other settings available to you through this Views Tools page. You can choose to **Add View signatures to all SQL queries** – doing this will add a VIEWS string to the WHERE clause in your query. This can cause a performance issue potentially though, as it is more work for the database. So, you can opt out of this. Drupal advises that you should only use this method for troubleshooting and development purposes.

You can also choose to **disable Views caching**. However, we do want Views to cache data because it will give us a performance enhancement. We can easily clear our Views cache using the above method if necessary, so let's leave the caching functionality for Views enabled.

If you do not have the Advanced Help module installed or if it's disabled, you can tell Views to ignore this and not show you errors telling you the Advanced Help integration with Views is disabled. Again, this is optional, but a nice setting to have if you do not want to see the error messages popping up.

We can also enable Views performance statistics if we have the Devel module installed. Go ahead and make sure this is checked. It should be, if you have been receiving Devel module Views statements regarding build times as noted earlier.

Set the **Page region to output performance statistics** as the **Footer** region – this is the default.

Add Views signature to all SQL queries (key=views_sql_signature, weight=0)
All Views-generated queries will include a special 'VIEWS' = 'VIEWS' string in the WHERE clause. This makes identifying Views queries in database server logs simpler, but should only be used when troubleshooting.

Disable views data caching (key=views_skip_cache, weight=0)
Views caches data about tables, modules and views available, to increase performance. By checking this box, Views will skip this cache and always rebuild this data when needed. This can have a serious performance impact on your site.

Ignore missing advanced help module (key=views_hide_help_message, weight=0)
Views uses the advanced help module to provide help text; if this module is not present Views will complain, unless this setting is checked.

Show query above live preview (key=views_ui_query_on_top, weight=0)
The live preview feature will show you the output of the view you're creating, as well as the view. Check here to show the query and other information above the view; leave this unchecked to show that information below the view.

Show other queries run during render during live preview (key=views_show_additional_queries, weight=0)
Drupal has the potential to run many queries while a view is being rendered. Checking this box will display every query run during view render as part of the live preview.

Do not show hover links over views (key=views_no_hover_links, weight=0)
To make it easier to administrate your views, Views provides 'hover' links to take you to the edit and export screen of a view whenever the view is used. This can be distracting on some themes, though; if it is problematic, you can turn it off here.

Enable views performance statistics via the Devel module (key=views-devel_output, weight=0)
Check this to enable some Views query and performance statistics if *Devel* is installed.

Disable javascript with Views (key=views_no_javascript, weight=0)
If you are having problems with the javascript, you can disable it here; the Views UI should degrade and still be usable without javascript, it just not as good.

Page region to output performance statistics (key=views-devel_region, weight=0):

Label for "Any" value on optional single-select exposed filters (key=views_exposed_filter_any_label, weight=0):

Click on the **Save configuration** button.

Using Panels caching

The Panels 3.x module allows you to distribute your site's content and Views into custom layouts. You can use Panels to set up and configure these layouts and then integrate content into the layout that you choose. The Panels module uses what the module developers call a pluggable caching mechanism. This mechanism allows you to set up caching for all of your panels as a whole or for specific individual content panes within one panel. This allows for a large amount of caching flexibility. This section assumes that you have some experience using the Panels module to configure a panel layout and add content to it. I'm going to walk you through those configuration steps quickly and then we'll look at Panels caching more closely.

Creating a panel and adding content to it

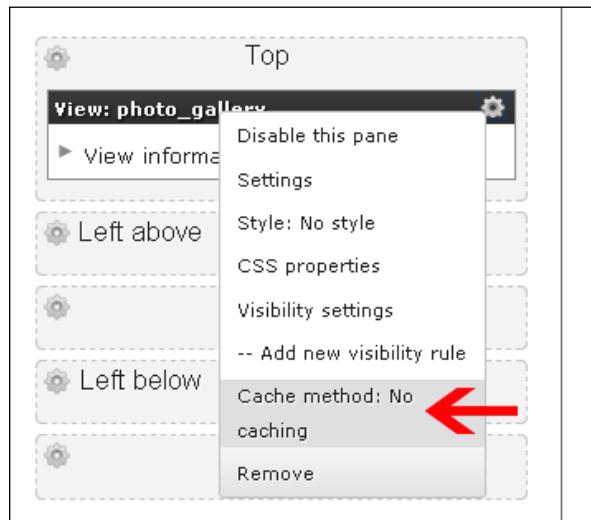
To get started with Panels caching, make sure you have a Panel to use on your site. I'm going to go ahead and set up a Panel quickly and add some content to it, which we'll use to test the Panels caching mechanism.

To create a new panel go to **Site building | Panels** and click on the **Panel page** link to create a new panel. Give the panel an **administrative title**, **machine name**, and a **path**. Click on the **Continue** button.

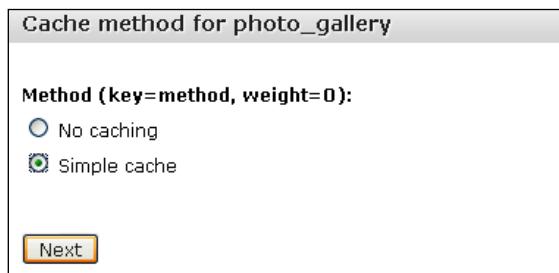
Choose a layout for your panel. Let's use the **Two column bricks** layout. Click on the **Continue** button.

This will launch your layout admin page where you can click on each panel section to add content to it. Add content to your panel's regions. We'll add two of our Views to the panel. Let's add our photo gallery View to our Top panel region. Click on the gear icon in the left corner of the region to open the region options. Click on **Add content**. Browse for your photo gallery view and add it to that panel region.

Once added, you will notice another gear icon in the upper right corner of the content pane you just added to your panel region. Click on this icon. When you click on this icon, a selection box will open and you will see a link to **Cache method**. The cache method for this content in your panel is currently set to **None**. Click on this link to open the cache method settings.



Similar to how you were presented with the Views caching configuration, Panels caching will ask you if you want to implement **No caching** on the content in the Panel or **Simple cache**. Select the **Simple cache** radio button and click on **Next**.



Like the Views caching, panels simple cache is time-based. Content will remain cached for the length of the time limit that you configure. Set a **Lifetime** cache time limit. We'll set a **Lifetime** cache of **1 hour**.

Also, set a **Granularity** cache if you are using contexts or arguments in your Panel. We're not using either, so we'll leave this set to **None**. If you do have contexts selected in your Panel, you can further cache depending on the context using this granularity cache mechanism. Again, Drupal and panels are giving you a lot of flexibility and granularity here.

Cache settings for photo_gallery Close Window

Simple caching is a time-based cache. This is a hard limit, and once cached it will remain that way until the time limit expires.

Lifetime (key=lifetime, weight=0):
1 hour

Granularity (key=granularity, weight=0):
None

If "arguments" are selected, this content will be cached per individual argument to the entire display; if "contexts" are selected, this content will be cached per unique context in the pane or display; if "neither" there will be only one cache for this pane.

Save

Click on the **Save** button to save your cache settings. Make sure to update and save your Panel again now that you have configured its cache mechanism by clicking on the **Update** button. Your panels are now set to cache the content you specifically configured using the cache configuration above.

Summary

In this chapter, we learned the following performance tips:

- Used the Throttle module to throttle our blocks and modules.
- Used the Devel module to generate test content, users, and categories.
- Configured caching for our Views and learned how to enable and disable caching per View. We also looked at how the Devel module returns information about our Views build times based on these cache settings.
- Learned how to clear our Views cache.
- Looked at how the Panels module uses caching and how you can enable caching per panel.

Let's take a break! When we come back in Chapter 5, we'll take a look at contributed modules that are built for specific types of performance monitoring. This will include the Boost, Memcache, DB Maintenance, Block Cache, Advanced Cache, and File Based Caching modules.



5

Using DB Maintenance and Boost

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

Using the DB Maintenance module

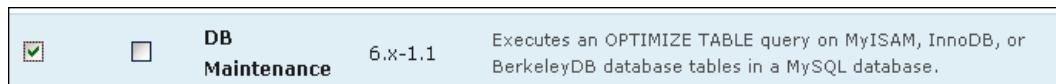
The DB Maintenance module can be used to optimize your MySQL database tables. Depending on the type of database you are running, the module allows you to use a function called `OPTIMIZE TABLE`, which troubleshoots and then optimizes various errors in your MySQL tables. For MyISAM tables, the `OPTIMIZE 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.

Home > Administer > Site configuration

DB maintenance

Executes an optimization query on database tables during cron runs. [\[more help..\]](#)

Log OPTIMIZE queries
If enabled, a watchdog entry will be made each time tables are optimized, containing information which tables were involved.

Optimize tables:
 Select how often database tables should be optimized. [Optimize now.](#)

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.

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  
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.

Home > Administer > Site configuration

DB maintenance

Executes an optimization query on database tables during cron runs.

[more help...]

Log OPTIMIZE queries
If enabled, a watchdog entry will be made each time tables are optimized, containing information which tables were involved.

Optimize tables:

Run during every cron ↗
Select how often database tables should be optimized. [Optimize now.](#) ↗



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.

- 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.

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

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:

Boost File Cache (key=boost, weight=0)

Boost - Static page cache (key=boost_enabled, weight=0):

- Disabled
- Enabled
- (Not Recommended) Set Boost & Core (if enabled) cache for each page

Static page caching is a mechanism that stores dynamically generated web pages as HTML files in a special cache directory located under the Drupal installation directory. By caching a web page in this manner, the web server can serve it out in the fastest possible manner, without invoking PHP or Drupal at all. While this does provide a significant performance and scalability boost, you should note that it could have negative usability side-effects unless your site is targeted at an audience consisting mostly of "anonymous" visitors.

Gzip page compression (Boost & Core) (key=page_compression, weight=0):

- Disabled
- Enabled

By default, Drupal compresses the pages it caches in order to save bandwidth and improve download times. This option should be disabled when using a webserver that performs compression.

Boost - HTML - Default minimum cache lifetime (key=boost_cache_lifetime, weight=0):

The minimum cache lifetime is the minimum amount of time that will elapse before the cache is emptied. Cache lifetime gets checked on cron runs.

Boost - XML - Default minimum cache lifetime (key=boost_cache_xml_lifetime, weight=0):

Enable the caching of this content type to enable this selection box.

Boost - JSON - Default minimum cache lifetime (key=boost_cache_json_lifetime, weight=0):

Enable the caching of this content type to enable this selection box.

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.

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:

The screenshot shows the 'Boost cacheability settings' configuration page. It includes sections for 'Cacheable settings (key=cacheability, weight=0)', 'Specific pages (key=boost_cacheability_option, weight=0)', and 'Pages (key=boost_cacheability_pages, weight=0)'. The 'Cacheable settings' section contains several checkboxes for different content types like URL Variables, HTML documents, XML/JSON feeds, CSS, and JS files. The 'Specific pages' section has three radio button options: 'Cache every page except the listed pages' (selected), 'Cache only the listed pages', and 'Cache pages for which the following PHP code returns TRUE (PHP-mode, experts only)'. The 'Pages' section is a large text input field with instructions below it about entering Drupal paths.

Boost cacheability settings (key=cacheability, weight=0)

Cache pages that contain URL Variables (key=boost_cache_query, weight=0)
Boost will cache pages that end with "?page=1" among others (anything with a "?" in the url).

Cache html documents/pages (key=boost_cache_html, weight=0)
Boost will cache most drupal pages.

Cache .xml & /feed (key=boost_cache_xml, weight=0)
Boost will cache .xml and /feed urls as xml data.

Cache ajax/json (key=boost_cache_json, weight=0)
Boost will cache ajax/json responses.

Cache .css (key=boost_cache_css, weight=0)
Boost will cache CSS files.

Cache .js (key=boost_cache_js, weight=0)
Boost will cache javascript files.

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 wild-card. Example paths are 'blog' for the blog page and 'blog/*' for every personal blog. <front> is the front page. If the PHP-mode is chosen, enter PHP code between <?php ?>. 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.

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:

The screenshot shows the 'Boost directories and file extensions (key=directories, weight=0)' configuration page. It includes fields for 'Cache Dir' (set to 'cache') and 'Cache file path' (set to 'cache/normal/variantcube.com/fire'). There is also a checkbox for 'Do not store the cache file path in the database'. Below the form, there are two expandable sections: 'Generated output storage (HTML, XML, AJAX)' and 'Static storage (CSS, JS)'.

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 single/shared 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)

Download at Wow! eBook

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:

```
# 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.

The resulting code looks like this:

```
### 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$">
    ForceType text/xml
</FilesMatch>
<FilesMatch "\.css\.gz$">
    ForceType text/css
</FilesMatch>
<FilesMatch "\.js\.gz$">
    ForceType text/javascript
</FilesMatch>

# Gzip Cookie Test
RewriteRule boost-gzip-cookie-test\.html cache/perm/boost-gzip-
cookie-test\.html\.gz [L,T=text/html]

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

```
# 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$">
```

```
    ForceType text/xml
</FilesMatch>
<FilesMatch "\.css\.gz$">
    ForceType text/css
</FilesMatch>
<FilesMatch "\.js\.gz$">
    ForceType text/javascript
</FilesMatch>
```

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

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.

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:

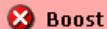
```
<!-- Page cached by Boost @ 2009-10-23 13:56:03, expires @ 2009-10-23  
14:56:03 -->
```

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.



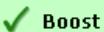
Boost

poormanscron is the wrong version

You need to get a newer version of [poormanscron](#), most likely the latest dev, or the 2.0 branch.

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.**



Boost

Boost Installed correctly, should be working if properly [configured](#).

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.



Clear ALL Boost cached data: 9 pages Clear Boost expired data: 2 pages

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

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:

The screenshot shows a configuration dialog for a Boost cache rule. At the top, there's a dropdown for 'Minimum cache lifetime' set to '1 hour'. Below it is a note 'Default: 1 hour'. The next section is 'Preemptive Cache' with a dropdown set to 'default'. Under 'Scope (key=selection, weight=0)', a dropdown is set to 'Page ID: 203'. A large 'Set Configuration' button is at the bottom of this section. Below the main configuration area, there are three checkboxes: 'Page ID (key=id, weight=0)' (unchecked), 'Content Type (key=type, weight=0)' (checked, with '- photo' listed under it), and 'Content Container (key=container, weight=0)' (checked, with '- node' listed under it). A 'Delete Configuration' button is located at the bottom of the list of checkboxes.

Minimum cache lifetime
(key=lifetime, weight=0):

1 hour

Default: 1 hour

Preemptive Cache
(key=push, weight=0):

default

Scope (key=selection, weight=0):

Page ID: 203

Set Configuration

Page ID (key=id, weight=0)
1 hour - 203

Content Type (key=type, weight=0)
- photo

Content Container (key=container, weight=0)
- node

Delete Configuration

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:

This page is being served **live** to
anonymous visitors, as it is not
currently in the static page cache.

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.

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.

6

Advanced Boost

In this chapter, we're going to continue to use the Boost module and look in detail at some of the module's advanced settings and its configuration.

By the end of this chapter you will know how to install, configure, and use the following contributed modules as well as how to use the advanced Boost functionality:

- Global Redirect module and configuration
- Transliteration module and configuration
- Configuring advanced Boost cache settings
- Using and configuring the Boost crawler
- Knowing where to look for the Boost .htaccess settings if you need to tweak their configuration after enabling Boost crawler settings

Updating contributed modules

Before we start looking at Boost's advanced settings, let's run our Status report and check on our site's contributed module status. I have been noticing some warning messages appearing across the site notifying us that we need to upgrade the following contributed modules to their latest security patch releases. So now is a good time to do that. The modules we're going to upgrade are:

- Devel module (upgrade to 6.x-1.18)
- FileField (upgrade to 6.x-3.2)

The screenshot shows the Drupal Status report interface. It lists three modules with update notifications:

- Devel 6.x-1.17**: Security update required! (Red background).
 - Security update: 6.x-1.18 (2009-Sep-23)
 - Download | Release notes

Includes: *Devel, Devel generate, Devel node access, Macro, Performance Logging*
- FCKeditor - WYSIWYG HTML editor 6.x-1.4-rc1**: Update available (Yellow background).
 - Recommended version: 6.x-1.4 (2009-Sep-14)
 - Also available: 6.x-2.0-rc2 (2009-Oct-26)
 - Download | Release notes

Includes: *FCKeditor*
- FileField 6.x-3.1**: Security update required! (Red background).
 - Security update: 6.x-3.2 (2009-Oct-20)
 - Download | Release notes

Includes: *FileField*

Download the updates from the modules' project page at <http://drupal.org/>, extract them locally on your desktop, and then upload the modules to your /sites/all/modules folder. You'll be asked to replace the existing files. Click on Yes to this and then browse to your Status report page once you have completed the copy of the module files. You should notice that your Update Notifications are enabled and in green mode showing that everything on your site module-wise has been updated. We are ready to move on.

Recommended modules that work with Boost

You will recall that when we installed the Boost module for the first time, there were a couple of recommended modules that work with Boost to help optimize your site for performance. These are the Global Redirect, Pathauto, and Transliteration modules. Let's briefly look at each of these modules and view their configuration and settings to understand how they will work on our site and help speed up performance.

Global Redirect

The Global Redirect module allows you to remove trailing slashes / from URLs. It also checks to see if your Drupal site has clean URLs enabled and, if so, will prevent unclean URLs from ever being accessed. You can do a lot more with this module and all of the details are listed on its Drupal project page here: <http://drupal.org/project/globalredirect>.

The module also redirects any specific node/*ID* page to its alias if an alias exists. This is important, as your site will get requests by visitors for duplicate nodes or pages. Some people will visit your node/12 page and others may visit it by its alias name. This module makes sure they can only load your node through its corresponding alias. This helps with performance and also makes sure you're not getting slammed by search engine bots. We already installed Global Redirect in the last chapter. Let's go to its settings page and see what we can configure. Go to **Site configuration | Global Redirect**. This will launch the configuration page.

Here you find the items you'll want to configure. First, make sure that you have selected to enable the deslash. This will remove all trailing / from URLs in your site. This will help to prevent duplicate node content from being loaded. One thing to consider is that if you do have a requirement for requests using a /, then you may need to leave this disabled. It will depend on your site requirements. Enable it for now.

Enable the **Non-clean to Clean URL** configuration. This will stop any requests loading a non-clean URL such as variantcube.com/fire?q=node/1.

Enable the **Remove Trailing Zero Argument** setting. This will help to trim your taxonomy URLs and prevent duplicate term URLs from loading when the URL refers to the same taxonomy term. For example, if you have a URL /taxonomy/term/1 and this term can also be loaded by going to /taxonomy/term/1/0, then enabling this functionality will trim the /0 from the URL and load the default depth of the term which is the /taxonomy/term/1.

Enable **Menu Access Checking**, as this will help to prevent your anonymous users or authenticated users who do not have permissions from trying to load admin content through admin URLs.

Also enable **Case Sensitive URL Checking**. This will make sure the user is always directed to the correct URL in your site based on specific characters in the URL.

Once you have configured the module settings, click on the **Save configuration** button.

Deslash:

Off
 On

If enabled, this option will remove the trailing slash from requests. This stops requests such as `example.com/node/1/` failing to match the corresponding alias and can cause duplicate content. On the other hand, if you require certain requests to have a trailing slash, this feature can cause problems so may need to be disabled.

Non-clean to Clean:

Off
 On

If enabled, this option will redirect from non-clean to clean URL (if Clean URL's are enabled). This will stop, for example, node 1 existing on both `example.com/node/1` AND `example.com?q=node/1`.

Remove Trailing Zero Argument:

Disabled
 Enabled for taxonomy term pages only
 Enabled for all pages

If enabled, any instance of "/0" will be trimmed from the right of the URL. This stops duplicate pages such as "taxonomy/term/1" and "taxonomy/term/1/0" where 0 is the default depth. There is an option of limiting this feature to taxonomy term pages ONLY or allowing it to effect any page. **By default this feature is disabled to avoid any unexpected behaviour**

Menu Access Checking:

Disabled
 Enabled

If enabled, the module will check the user has access to the page before redirecting. This helps to stop redirection on protected pages and avoids giving away secret URL's. **By default this feature is disabled to avoid any unexpected behaviour**

Case Sensitive URL Checking:

Disabled
 Enabled

If enabled, the module will compare the current URL to the alias stored in the system. If there are any differences in case then the user will be redirected to the correct URL.

[Save configuration](#) [Reset to defaults](#)

Transliteration and Pathauto

This module can be enabled from your main modules admin list. The current version is 6.x-2.1. Go ahead and enable it on your site. The module does not have a configuration admin page in your Drupal site, but you can learn more about its configuration and what it does through its Drupal project page: <http://drupal.org/project/transliteration>.

In a nutshell, this module provides transliteration resources for other Drupal modules including Boost, and also sanitizes your file upload names. For example, if a user uploads a file to your site using the Drupal attachment core module and the filename contains special non-ASCII characters, this module, if enabled, will convert and try to represent that data in US-ASCII characters, which are the universally accepted character codebase.

You can also use Transliteration along with the Pathauto module and it works similarly to its configuration for file uploads. In your Pathauto settings, you can enable Transliteration prior to creating any automatic alias. In order to do this you'll need to add an `i18n-ascii.txt` file to the Pathauto directory. You can view instructions for configuring this functionality by going to **Site building | URL aliases | Automated alias settings** and then expanding the General settings in the Pathauto module configuration. You will see a checkbox next to **Transliterate prior to creating alias**.

<input checked="" type="checkbox"/> Transliterate prior to creating alias
When a pattern includes certain characters (such as those with accents) should Pathauto attempt to transliterate them into the ASCII-96 alphabet? Transliteration is determined by the <code>i18n-ascii.txt</code> file in the Pathauto directory.

You can find this `i18n-ascii.txt` file by doing a search for it on <http://drupal.org/>. Using Google search, one example of the file is here: <http://textpattern.googlecode.com/svn/development/crockery/textpattern/lib/i18n-ascii.txt>. Download the file and copy it to your Pathauto module directory. Once you do this, you can enable the **Transliterate prior to creating alias** setting. When you upload this file, make sure to rename it `i18n-ascii.txt` so that it's recognized by the Pathauto and Transliteration modules. Again, much of this will be your personal preference and decision as to whether you want to use the Transliteration module in your site. If you do, it will open up various configuration possibilities for your Boost module. It's good to understand generally how all of these modules work hand in hand to help optimize your site.

Let's take a look at some advanced Boost module settings and their configuration.

Advanced Boost settings

Let's do a quick review of our Boost configuration upto this point. We have set up Boost to cache our Drupal nodes. This stores our Drupal nodes as static HTML web pages. When an anonymous user visits our site, they are delivered the static HTML version of the web page, and this helps considerably with speed and optimization of our page loads. We have set our minimum cache lifetimes for HTML, XML, and JSON content to be 1 hour. We also have the option of clearing all of our Boost-cached data and expired data manually by clicking on the respective buttons on the Boost settings page in the Boost File Cache pane.

We configured Boost to cache pages that contain URL variables even though we're using clean URLs on our site. We are caching all HTML documents or Drupal pages, all XML and feed content, all AJAX and JSON content, and CSS and JavaScript files. We also determined that we can tell Boost to only cache specific pages or use PHP code to run conditional statements for our cache settings. This is very similar to how the Blocks settings and configuration works.

We configured Boost to store all of our cached files and pages in the main /cache directory and specifically in /cache/normal/variancube.com/fire. Once we configured this, we edited our .htaccess file to permit this storage to occur. We also looked at the specific configurations we can set up to store files using Boost. We can store our HTML, XML, and AJAX files in specific subdirectories in the /cache folder, and we can also store our CSS and JS files in specific subdirectories. Boost provides a lot of flexible storage options for us to take advantage of.

Boost advanced settings

Let's start looking at our advanced Boost configuration. The first item we want to note is that we can tell Boost to clear expired pages on cron runs. This will clear all expired cached pages when our cron runs. If you disable this, you will need to clear your Boost cache manually through the **Boost: Pages cache status block** (we looked at this block configuration in Chapter 5).

The next setting is even more flexible. Boost will only clear and flush expired content if it sees that there's been a change in the database timestamps, that is, if there's new database content. This is a setting you'll want to enable if you do not want to rely on the cron run to clear your site cache and if you have content being posted to your site frequently that you want your anonymous users to view without problems. Here we will take advantage of the Boost timestamp function, and use it to rebuild and flush cache only on pages that have been updated in the database. This PHP-based functionality will refresh the cache on that one specific Drupal node and not have to flush or rebuild the entire site cache. This enables you to target your Drupal nodes and database much more granularly rather than having to flush the entire Boost cache.

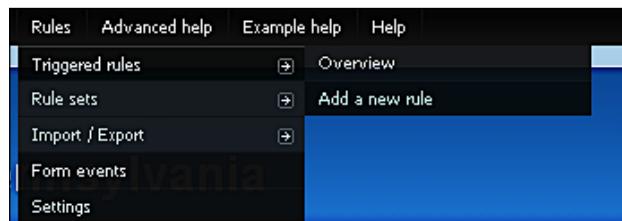
Let's go ahead and do this. We'll first check the box next to the **Check database timestamps for any site changes**.

Next, we're going to install the Rules module so that we can set up a rule that will run cron on our site every time a node is updated and saved. The Rules module allows you to define conditional PHP statements throughout your Drupal site and on specific Drupal events. The Rules module is on drupal.org (<http://drupal.org/project/rules>). Download the latest version of Rules, which is 6.x-1.1. Extract and upload the module to your /sites/all/modules directory.

Once uploaded to your site, go to your modules admin list and enable the Rules module and its constituent modules. Also, enable the PHP filter module if you have not already.

Enabled	Throttle	Name	Version	Description
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Rules	6.x-1.1	Lets you define conditionally executed actions based on occurring events. Required by: Rules Administration UI (disabled), Rules Forms support (disabled), Rules Scheduler (disabled), Rules Simpletest (disabled)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Rules Administration UI	6.x-1.1	Provides the administration UI for rules. Depends on: Rules (disabled)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Rules Forms support	6.x-1.1	Provides events, conditions and actions for rule-based form customization. Depends on: Rules (disabled)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Rules Scheduler	6.x-1.1	Schedule the execution of rule sets. Depends on: Rules (disabled)
<input type="checkbox"/>	<input type="checkbox"/>	Rules Simpletest	6.x-1.1	Tests the functionality of the rule engine Depends on: Simpletest (missing), Rules (disabled)

Save your module configuration. Then visit the Rules configuration page by going to **Rules | Triggered Rules | Add a new rule**.



We're going to add a new rule and event by doing the following:

1. In the **Label** field, type in the following: **Run cron when node is saved**.
2. In the **Event** drop-down menu, choose the following event under the Node category: **Content is going to be saved**.

3. Check the box next to the statement: **This rule is active and should be evaluated when the associated event occurs.**
4. Click on **Save changes**.

The screenshot shows the 'Triggered rules' configuration interface. The 'Rule settings' tab is selected. The form includes fields for 'Label' (Run cron when node is saved), 'Event' (Content is going to be saved), 'Categories' (empty), 'Weight' (0), and a checked checkbox for 'This rule is active and should be evaluated when the associated event occurs'. A 'Save changes' button is at the bottom.

Once you click on **Save changes**, you'll be presented with a screen that allows you to edit the **Database Timestamp Boost** rule you just configured. Here do the following:

- Click on the **Add an action** link.

The screenshot shows the 'Editing rule Database Timestamp Boost' configuration interface. It displays a success message: 'The rule Database Timestamp Boost has been added. You can start'. Under 'Rule elements (key=elements, weight=0)', it shows 'ON event Content is going to be saved' and 'DO'. A red arrow points to the '+ Add an action' link.

- From the **Select an action to add** drop-down menu, select the **Execute custom PHP code** statement and then click on **Next**.

Add

Select an action to add (key=name, weight=0): *

Clear a page from the boost cache.

Publish content
Remove content from front page
Save a content
Set content title
Set the content author
Unpublish content
Unpublish content containing keyword(s)

PHP

Execute custom PHP code

Path

Create or delete a content's URL alias
Create or delete an URL alias

Rule Scheduler

Delete scheduled rule sets
Schedule Example: Empty rule set working with content

Rule Sets

Example: Empty rule set working with content

Rules

Add a new date variable
Add a new number variable

- On the next screen, in the **PHP Code** box, copy and paste the following code. Remember here that you do not need to include the opening and closing PHP brackets <?php and ?>.

```
global $base_root, $base_path;
drupal_http_request($base_root . $base_path . 'cron.php');
```

PHP Code: *

```
global $base_root, $base_path;
drupal_http_request($base_root . $base_path . 'cron.php');
```

The code that should be executed. Don't include <?php ?> delimiters.

Weight:

0

Adjust the weight to customize the ordering of actions.

Save **Delete**



This code is provided from the Boost advanced settings help and support documentation page on drupal.org (<http://drupal.org/node/583264>)



- Click on **Save**. You will receive a confirmation that **The action Execute custom PHP code has been saved**.
- Return to your Boost advanced settings and check the box next to **Check database timestamps for any site changes**. Only if there's been a change will Boost flush the expired content on cron.
- Save your Boost settings configuration.

Check database timestamps for any site changes. Only if theres been a change will boost flush the expired content on cron.

As soon as you do this, Boost will now refresh the cache on any one page on your site that has been changed in the database.

Testing your Database timestamp settings

Let's go ahead and test the functionality we just implemented. You'll see immediately how this works in the new caching mechanism we've configured. First, logout of your site and then browse to a page and load it as an anonymous user. Check your /cache/normal/variantcube.com/fire/node folder to make sure the HTML version of the cached page is showing up. I'm visiting node/207 on my site and I notice immediately that once I load node/207, the HTML version shows up in my cache folder.

If you are having trouble seeing this happen through FTP or SFTP, refresh your remote window in your FTP client.

Now log back into your site and visit the same node as your admin authenticated user. Edit the node and add some content to it, or tweak some content so that the change will be noticeable to your anonymous users. I'm editing node/207 and I'm adding some new textual content to my title field for that node. Save your node.

Engine 2 - Back of the Mack

[View](#) [Edit](#) [Outline](#) [Track](#) [Dev load](#) [Dev render](#)

Photo Gallery *Engine 2 - Back of the Mack* has been updated.

Submitted by **admin** on Sun, 07/26/2009 - 02:30

Photo:



Now, immediately go into your FTP client and refresh your remote view. You should see the HTML version of the node disappear. The script you are running through the Rules module and Boost is working. You made a database level change to this node and now the node cache has been flushed. The HTML version is gone and you're ready to serve out the latest version of this node to your anonymous users. To finish the test, logout of your site and load the same node, as an anonymous user. Now, immediately go into your FTP client and refresh. You should now see the cached version of the node again and you should also see the edited content on your node as an anonymous user. It's working!

You'll also notice that all of your other cached HTML page versions are unchanged and unaffected by this rule's actions. The PHP script you are running only targeted this one node that had changed content. Bear in mind that running cron using the rule you have created will cause the cron to run every time a node is saved. This may cause cron to run more frequently than you have it set to run via your Poormanscron module or a cron configuration you have configured on the server. If this is the case, then your cron could take more time to run because it will be flushing other module actions, for example, if your modules rely on cron for specific module functionality. Just keep this in mind as you work with this rule.

Let's look at some more Boost advanced settings. The next checkbox on the Boost settings page asks you if you want to force the site to only allow ASCII characters in a URL path. I would suggest leaving this checked if you're not using any content translation or i18n functionality on your Drupal site.

The next setting pertains to multisite environment and development. If you're running a multisite, you can select to **Flush all sites caches in this database**. This will only work if you're running a multisite environment from one Drupal database. So, if you want to flush all the sites' caches, you can enable this here. We're not running multisite currently, so I'm going to leave this unchecked.

Asynchronous Operation: output HTML, close connection, then store static file helps to speed up your caching process even more. Here the caching will generate the cached HTML page without paying close attention to the PHP on the page. PHP will remain running in the background, but the cache page will only pay attention to HTML. This will also help to generate faster page loads.

The next setting allows you to not only clear cached files, but also the entire folder in your /cache directory. This gives you an added feature to help clean house, but you need to be careful if you have set specific writeable subdirectories. If you clear those subdirectories, you will need to rebuild them again through FTP or a file manager and set the correct permission because clearing them will wipe them from the server. So use this with caution.

Only allow ASCII characters in path (key=boost_only_ascii_path, weight=0)

Only allowing ACSII characters is a safe way to cache pages. It severely limits i18n support so this can be turned off. Fair warning, disabling this may cause "page not found" errors depending on your url structure (spaces are bad, ect...). If you follow RFC 3986 you should be ok.

Flush all sites caches in this database (singe db, multisite). (key=boost_flush_all_multisite, weight=0)

This will flush/expire all cached files stored in this database, instead of only being specific to this site. Useful for i18n sites. You need to setup a separate cron call for each database (in your multisite install) either way though. This covers shared database usage; or place the boost tables into the a shared database, to have this setting work in a multiple database environment.

Asynchronous Operation: output HTML, close connection, then store static file.

(key=boost_asynchronous_output, weight=0)

Run php in the background. When a cached page is generated, this will allow for faster page generation; downside is the headers are not the standard ones outputted by drupal; sends "Connection: close" instead of "Connection: Keep-Alive".

Clear all empty folders from cache. (key=boost_flush_dir, weight=0)

Disable this if you have to set settings for each dir/subdir, due to the way your server operates (permissions, etc...).

The next four settings refer to Boost's integration and behavior with the CCK, Views, and Taxonomy modules. You can tell Boost to **Clear all cached pages referenced via CCK with a node on insert/update/delete**. This will clear the cached pages for any node that references another node via a CCK node reference field. Both nodes will be cleared. There is a module that works well with this functionality called the **Node Referrer** module. If you plan to use this functionality, you should install and configure this module.

You can also have Boost **Clear all cached terms pages associated with a node on insert/ update/delete**, **Clear all cached views pages associated with a node on insert/update/and delete**. Note that if you do this specifically, the views page associated with a node on insert can be a slow process because all of your Views will process so that Boost can locate the node that is being cached. Use this functionality with caution.

<input checked="" type="checkbox"/> Clear all cached pages referenced via CCK with a node on insert/update/delete (key=boost_flush_cck_references, weight=0) The Node Referrer module is recommended.
<input checked="" type="checkbox"/> Clear all cached terms pages associated with a node on insert/update/delete (key=boost_flush_node_terms, weight=0) Works with view's taxonomy/term/% path as well as core.
<input checked="" type="checkbox"/> Clear all cached views pages associated with a node on update/delete (key=boost_flush_views, weight=0)
<input checked="" type="checkbox"/> Clear all cached views pages associated with a node on insert (key=boost_flush_views_insert, weight=0) WARNING: This can be very slow, all views get run to find out where this node lives.

You can tell Boost to clear cache when you put your site offline for maintenance mode, overwrite the cached file if it already exists, and not to cache a page if there is a PHP error on the page. We'll go ahead and check this box. You can also check the box next to the **Do not cache if a message is on the page**.

You can choose to turn off clean URLs for logged in users. We want to use clean URLs for both of our user bases, so we'll leave this unchecked. This setting is recommended for sites that have mostly admin-based authenticated users who may not mind using non-clean URLs.

You can also choose to use Aggressive Gzip caching if you have your site configured to use Books for Gzip caching.

<input type="checkbox"/> Clear Boosts cache when site goes offline (key=boost_clear_cache_offline, weight=0) Under site maintenance when the status is set to offline, boost clears its cache. If you do not want this to happen, clear this checkbox. Pages that are not cached will still send out a Site off-line message, so be smart if turning this off.
<input type="checkbox"/> Overwrite the cached file if it already exists (key=boost_overwrite_file, weight=0) This is useful if crawling a site before it goes live.
<input checked="" type="checkbox"/> Do not cache if php error on page (key=boost_halt_on_errors, weight=0) Selected - Do not cache the page if there are PHP errors. Not Selected - Cache pages even if it might contain errors.
<input checked="" type="checkbox"/> Do not cache if a message is on the page (key=boost_halt_on_messages, weight=0) Selected - Do not cache the page if there are drupal messages. Not Selected - Cache pages even if it might contain a drupal message.
<input type="checkbox"/> Turn off clean url's for logged in users (key=boost_disable_clean_url, weight=0) Drupal will output non clean url's for non anonymous users. This allows for the browser to cache the page and still have logging in work. This is more on the extreme side of tweaks.
<input checked="" type="checkbox"/> Aggressive Gzip: Deliver gzipped content independent of the request header. (key=boost_aggressive_gzip, weight=0) In order to deliver gzipped content independent of the header, this will test for gzip compression in a small iframe by sending it compressed content. This compressed content is javascript which creates a cookie with a note of gzip support. On the server side it checks for the cookie and then sends out gzipped content accordingly. See Website Performance - Activate Gzip . In short some firewalls/proxies mangle the gzip header; this gets around that. iframe is on non compressed version of the frontpage only.
Files: Enter in a 4 digit number (octal) that will be used by chmod() . Example 0664 (key=boost_permissions_file, weight=0): <input type="text"/>
Sometimes because of funky servers you need it use a different file mode than the default.
Directories: Enter in a 4 digit number (octal) that will be used by chmod() . Example 0775 (key=boost_permissions_dir, weight=0): <input type="text"/>

Another one of the tweaks that Boost allows for is the **Expire content in DB, do not flush file**. This will expire the database entry for the file or node before actually flushing the cached version of the node. Again, this provides even greater flexibility in your Boost configuration.

Finally, you can tell Boost to **Ignore cache flushing** (recommended only if you are caching CSS and JS files) and whether you want to record all Boost errors to the Drupal watchdog recent log entries.

Ignore cache flushing (key=boost_ignore_flush, weight=0): <input checked="" type="radio"/> Disabled <input checked="" type="radio"/> Only Ignore Clear Entire Cache Commands (Recommended if caching css/js files) <input type="radio"/> Ignore Clear Entire Cache Commands & Cron Expiration <input type="radio"/> Ignore All Delete Commands (Not Recommended) Make a selection to put your site into a static cached state. Recommend turning on CSS & JS caching if enabled.
Watchdog Verbose Setting (key=boost_verbose, weight=0): <input checked="" type="checkbox"/> 5 Record all errors to the db log (watchdog) <input type="checkbox"/>

Boost crawler settings

Boost crawler allows you to cache your Drupal pages preemptively, allowing pages to be cached before anonymous users or any visitor on your site loads them. First, you need to check the box to enable the cron crawler. You must check this box and save your entire configuration before you are able to check the specific caching boxes in the Boost crawler section of the Boost settings. I encourage you to test this functionality on your site before putting it in a production environment. You may not need to use pre-caching, but again the module provides you with this flexible functionality. The handbook content recommends using this functionality on shared hosts and in conjunction with the database timestamp configuration we enabled earlier in this chapter. You may also have to update your .htaccess settings to use this type of configuration.

Boost crawler (key=crawler, weight=0)

Enable the cron crawler (key=boost_crawl_on_cron, weight=0)
Pre-cache boosted URL's so they get cached before anyone accesses them. Enable the crawler first and save settings to use Preemptive Cache settings.

Do not flush expired content on cron run, instead recrawl and overwrite it. (key=boost_loopback_bypass, weight=0)
The "Overwrite the cached file if it already exists" must be turned on in order to enable this.

Preemptive Cache HTML (key=boost_push_html, weight=0)
Crawl Site after cron runs, so the cache is primed.

Preemptive Cache XML (key=boost_push_xml, weight=0)
Crawl Site after cron runs, so the cache is primed.

Preemptive Cache AJAX/JSON (key=boost_push_json, weight=0)
Crawl Site after cron runs, so the cache is primed.

Crawl All URL's in the url_alias table (key=boost_crawl_url_alias, weight=0)
Preemptively cache all urls found in the Drupal url_alias table. This will crawl that page even if it is not expired. Enable & run cron to get the boost cache loaded.

Crawler Throttle (key=boost_crawler_throttle, weight=0):

Wait X micro seconds in between hitting each url. 1000000 is 1 second.

Crawler Batch Size (key=boost_crawler_batch_size, weight=0):

Number of URL's each thread grabs per database operation.

Number Of Threads (key=boost_crawler_threads, weight=0):

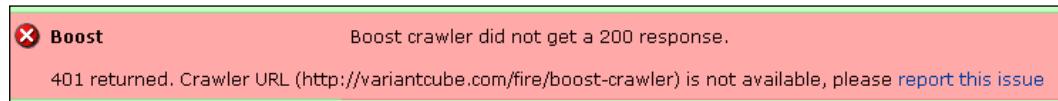
Be careful when choosing more than 2 threads.

Reset Crawler & Cron Semaphore

Estimated Time to crawl site - highly inaccurate (key=boost_crawler_eta, weight=0):

Remember that if you do change the above settings in the Boost crawler section, you may need to regenerate your .htaccess rules and re-paste them into your .htaccess file, as you will now have updated and potentially changed the settings. So, save your entire Boost advanced settings and main settings configuration first. Then, having done this, run your site's Status report. This will tell you whether Boost is configured correctly or if you need to update your .htaccess.

For example, I enabled the Boost cron crawler. Once I did this and visited my Status report, I received this error message:



If you do need to make updates, go back to your Boost configuration page and click on the **Boost htaccess rules generation** link and copy your latest Generated Rules. Then paste this into your .htaccess file to update it.

Since I'm receiving this error, I'm going to go back and regenerate my Boost htaccess rules and re-paste them into my .htaccess file. Once I do this and run my Status report again, I should receive the green box telling me that Boost is running correctly and can run the Boost crawler.

Summary

In this chapter, we learned the following Boost module advanced configuration tips:

- Installing and configuring the Global Redirect contributed module.
- Setting up a Boost database timestamp PHP-based rules configuration for clearing static pages using the Rules module and Boost advanced configuration
- Testing the database timestamp rule
- Configuring additional Boost advanced configuration
- Configuring Boost crawler settings

Let's take a break! When we come back in Chapter 7, we'll take a look at contributed modules that are built for specific types of performance monitoring. These will include the Memcache, Advanced Cache, and File Based Caching modules.

[Download at Wow! eBook](#)

7

Using Memcache API and Integration

In this chapter, we're going to return to our discussion of Drupal caching mechanisms and take a detailed look at the Memcache API and Integration contributed module along with the best methods of installing and configuring this module to allow for more granular and advanced cache configurations within our site. In Chapters 5 and 6, we used the Boost module to enable advanced caching for our anonymous site visitors. In this chapter and in Chapter 8, we're going to look at the best methods of enabling caching for our authenticated users, and investigate how to get even more control and flexibility over the Drupal caching system using contributed modules.

By the end of this chapter you will know how to:

- Install a development WAMP environment using MoWeS Portable
- Install Memcached binary libraries on your development server
- Integrate and configure Memcached to work with your PHP 5.2.x application
- Extract and install the Memcache API and Integration module
- Configure your `settings.php` to recognize and use the Memcache API module
- Enable the Memcache admin module
- Test the module on your development site and view its detailed reporting and statistics

The focus of this chapter will be how to use the Memcache API and Integration contributed caching module and integrate this with a Drupal 6.x site, and how best to implement these modules first in a development or sandbox type of hosting environment with the flexibility to implement them later on production websites. By doing this we'll actually see how the modules can contribute to performance optimization and speed on our sandbox and development sites before rolling them out into a production environment. Let's get started!

Using the Memcache API and Integration module

The Memcache API and integration module provides an API for integration with the Memcached daemon or service. There are many methods of installing the prerequisites for using this module, specifically the Memcached library; but we're going to focus on how to easily install Memcached and the Drupal 6.x version of the module in a Windows WAMP environment using PHP 5.2.x. In order to use the Drupal module, we'll need to first install the Memcached service on our local development server and then integrate this service with our PHP version. There are many instructions available on the web for installing Memcached on a Linux or Mac OS system. You can do a Google search for tutorials on how to get Memcached installed or see the note below with links to resources on <http://drupal.org/> and on Lullabot's website. Our focus here is to get Memcached up and running as quickly as possible so that you can see examples of how it works in a sandbox environment, so we're going to install using WAMP on Windows XP.



The drupal.org project page for the Memcache API and Integration module is here and provides a wealth of information and resources about the module: <http://drupal.org/project/memcache>. There are instructions and tutorials on how to install Memcached and the module in a Linux environment (Red Hat, CentOS, and Debian) through the Lullabot.com websites at the following URLs: http://www.lullabot.com/articles/how_install_memcache_debian_etch and http://www.lullabot.com/articles/installing_memcached-redhat-or-centos. If you want to read more about Memcached, you can find resources on its main organization website here: <http://memcached.org/>.

MoWeS Portable development WAMP server

The first thing you'll want to do in order to follow these instructions is make sure you're running a WAMP localhost environment on your Windows XP system. I am using the portable application called MoWeS that installs Apache, MySQL, and PHP on your C drive so that you can run the Apache web server in a development configuration through your localhost. You can download the MoWeS Portable packages here: <http://www.chsoftware.net/en/useware/mowes/mowes.htm>.

To install MoWeS follow these steps:

1. Visit the MoWeS website and go to the **Download** link.
2. Select the **I do not have a MoWeS portable II Package and want to obtain a new package** link if you are installing a brand new MoWeS install. Click on **Go**.
3. Choose the applications you want to install. I chose the following: **Apache 2, MySQL 5, PHP52 (version 5.2.10)**. If you choose to use another popular localhost package such as **WAMP (Windows Apache, MySQL, PHP)** make sure to use the package that includes PHP 5.2.x. PHP 5.3.x will cause issues with some Drupal contributed modules due to an issue with `ereg()` being deprecated in PHP 5.3 and higher. So for now make sure that you are using PHP 5.2 in your production and live environments.
4. You do not need to choose any application software because you can install Drupal manually in your `/www` directory once you install MoWeS.
5. Download the package called `mowes_portable.zip` to your desktop.
6. Move the ZIP file to your C drive and extract it. It will create all of the application directories necessary including `mysql` and `php5`.
7. Run the `mowes.exe` file to install MoWeS. If you run into installation issues, check the MoWeS documentation and support on the MoWeS site.

Once you get MoWeS (or a similar portable application that provides a WAMP environment) installed, you should have the following folders located in your C drive:

- `www`: This is where all of your localhost Drupal sites will reside.
- `apache2`: The Apache web server folder and files.
- `mysql`: The MySQL folder and files.
- `php5`: PHP folder and files.
- `mowes.exe`: This is the MoWeS Portable executable file that lives at the root of your C drive. Double-click on this to fire up your MoWeS Portable application and start the Apache web server.

Start up your MoWeS Portable application by clicking on the `mowes.exe` file. Once you do this, you should see a MoWeS Portable status box open telling you that Apache and MySQL are running without issue. Everything in the box should be green and you should also see a **Stop server** button and an **End** button.



Now that you have a MoWeS development environment make sure you install a Drupal site and database that you can use for the rest of this chapter and its examples, following the instructions provided in Chapter 1 of this book for installing Drupal. Once you have a Drupal site installed and configured, you're ready to install Memcached and the Memcache API.

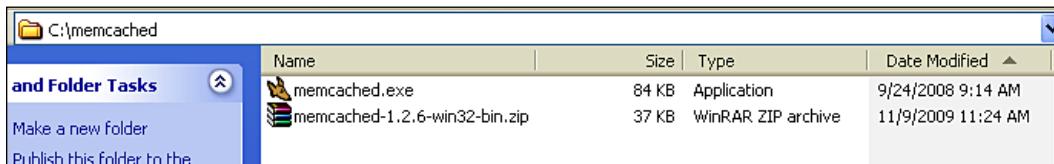
Installing Memcached libraries and service

Memcached is an entire service similar to MySQL or PHP that runs on your web server and integrates with other applications like PHP and Drupal. We need to get Memcached installed first on our development server and then we need to integrate it with our PHP installation and environment.

To install Memcached on Windows XP, you first need to download the `memcached-1.2.6` version for `win32`. I'm using the binary version available from this website: <http://code.jellycan.com/memcached/>.

Download the `memcached-1.2.6-win32-bin.zip` to your C drive.

We're going to now install Memcached as a service on our web server. To do this, first create a new folder on your C drive called memcached. You should have a folder C:\memcached and in that folder move and extract the ZIP file you just downloaded. This will extract and create a file called memcached.exe. You should see something in your folder that looks like this:



You now have the executable file of the Memcached binaries on your local computer. Now we want to install them. To do this, follow these steps:

1. Click on your **Start** menu button, select **Run**, then enter cmd or cmd.exe, and hit the *Enter* key. This will put you in command-line mode.
2. Install the Memcached service by typing the following:
c:\memcached\memcached.exe -d install
3. Once installed, you need to start the service. To do this type in the following:
c:\memcached\memcached.exe -d start

The Memcached service should now be installed and started. Now we need to integrate it with your PHP so that you can use it with your Drupal websites.

Integrating and testing Memcached with PHP 5.2.x

The first thing you need to do is check your PHP extensions directory to see if the Memcached extension is already installed. Go to c:\php5\ext directory and see if you can locate the php_memcache.dll file. If it's not there, you need to download it and paste it into your \ext folder.

You can get a copy of the DLL file here. This file works with PHP 5.2.x configurations: www.pureformsolutions.com/pureform.wordpress.com/2008/06/17/php_memcache.dll.

Now you need to tell your main php.ini file to reference the php_memcache.dll extension. To do this, find your php.ini file and open it to edit in WordPad. Look for the extensions section of the .ini file and add the following extension:

```
extension=php_memcache.dll
```

Save your `php.ini` file and then click on the **Stop server** button in your MoWeS Portable utility window. Then **Restart** the server. Once you restart Apache, go ahead and create a file called `phpinfo.php` and add this line of code to it: `<?php phpinfo(); ?>`. Then browse to that `phpinfo` file in your web browser, for example: `http://localhost/start/phpinfo.php`. This will give you all of your PHP configuration and lists of extensions. You should see the following section on this page showing you that Memcache is indeed installed and running on the server:

memcache		
memcache support	enabled	
Active persistent connections	0	
Version	2.2.4-dev	
Revision	\$Revision: 1.99 \$	
Directive	Local Value	Master Value
<code>memcache.allow_failover</code>	1	1
<code>memcache.chunk_size</code>	8192	8192
<code>memcache.default_port</code>	11211	11211
<code>memcache.hash_function</code>	crc32	crc32
<code>memcache.hash_strategy</code>	standard	standard
<code>memcache.max_failover_attempts</code>	20	20

You can also test your Memcached installation by creating a PHP file in your `/www` root and running the following code. I called my file `memcache.php` and the code follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US">
<head>
    <title>Memcache Test</title>
    <meta http-equiv="content-type" content="text/html;
          charset=iso-8859-1" />
    <title>Testing Memcache</title>
</head>
<body>
<?php
    $memcache = new Memcache;
    $memcache->connect("localhost",11211);
    # You might need to set "localhost" to "127.0.0.1"
    echo "Server's version: " . $memcache->getVersion() . "<br />\n";
    $tmp_object = new stdClass;
```

```
$tmp_object->str_attr = "test";
$tmp_object->int_attr = 123;
$memcache->set("key",$tmp_object,false,10);
echo "Store data in the cache (data will expire in 10 seconds)
<br />\n";
echo "Data from the cache:<br />\n";
var_dump($memcache->get("key"));
?>
</body>
</html>
```

Running this code should return something similar to the following as a web page display when you browse to `/localhost/memcache.php`. If you receive no errors, you're good to go and have installed Memcached correctly.

```
Server's version: 1.2.6
Store data in the cache (data will expire in 10 seconds)
Data from the cache:
object(stdClass)#3 (2) { ["str_attr"]=> string(4) "test" ["int_
attr"]=> int(123) }
```

Installing the Memcache API and Integration module

We're now ready to install the Drupal contributed Memcache API module into our Drupal site. Bear in mind here that this is a good starting point for you to work from. But the focus here will be on a Windows development environment using the MoWeS Portable Apache configuration. You may be using a different WAMP set up on Windows or using MAMP on a Mac. Your configuration may be different, so you can use this as a starting guide to getting Memcache up and running.

There is an additional configuration step to take with the previously installed Memcached libraries before we continue. Open up your `php.ini` file in WordPad again and add the following line if it's not in there already:

```
memcache.hash_strategy=consistent
```

You can add this line immediately after the last extension listed in your Windows extensions section.

If you run `phpinfo.php` before making this change, you'll notice that your `memcache.hash_strategy` is set to standard—you're changing it to be consistent. You'll need to restart your Apache server using MoWeS to effect the changes. Go ahead and do this and then refresh your `phpinfo` page.

<code>memcache.hash_strategy</code>	consistent	consistent
-------------------------------------	------------	------------

Now you're ready to install and enable the module. First, you want to put your development Drupal site in offline mode. This is good practice before implementing any advanced modules such as Memcache.

Before installing the Memcache module make sure to read thoroughly the installation instructions provided on the `drupal.org` project page as well as those in the `INSTALLATION.txt` file that comes with the module.

Here are the steps to install the Memcache API and Integration module:

- Put your Drupal site in offline mode.
- Download the latest version of the module from the project page here: <http://drupal.org/project/memcache>. Latest version is 6.x-1.4.
- Extract and copy the `memcache` module folder to your `/sites/all/modules` folder.
- Confirm that you have Memcached running as a service on your server (using instructions in the previous section).

Now, as you're going to be using the Memcache module to perform all caching on your website, you need to make sure that your Drupal site knows which Memcache include file to use when it's running its caching mechanism.

There are two Memcache module INC files: `Memcache.inc` is the default core INC file to use whereas `Memcache.db.inc` can be used as a variant that offers you more flexibility. Using `Memcache.db.inc` will save all your cached data into the MySQL database, so you can still use caching if your site is offline. `Memcache.inc` does not store any data in the database, so it's much lower overhead and lighter in weight on your site and site database, and this will give you the best performance results. However, this file also uses the most memory resources allocated to your site, so make sure you have enough PHP memory running because it will be storing all the page caching directly in memory and not in the database.

You can decide which INC files to use. You do need to tell Drupal to use a specific INC file, so you'll need to edit your `settings.php` file and add the following code to call the specific `.inc` file you want to use. We'll use the `memcache.db.inc` file for now and then tweak this to try and run everything from memory using the `memcache.inc` file later. Update the `$conf` in your `settings.php` file to contain the following code:

```
<?php
$conf = array(
    // The path to wherever memcache.inc is. The easiest is to simply
    point it
    // to the copy in your module's directory.
    // 'cache_inc' => './sites/all/modules/memcache/memcache.inc',
    // or
    'cache_inc' => './sites/all/modules/memcache/memcache.db.inc',
);
?>
```

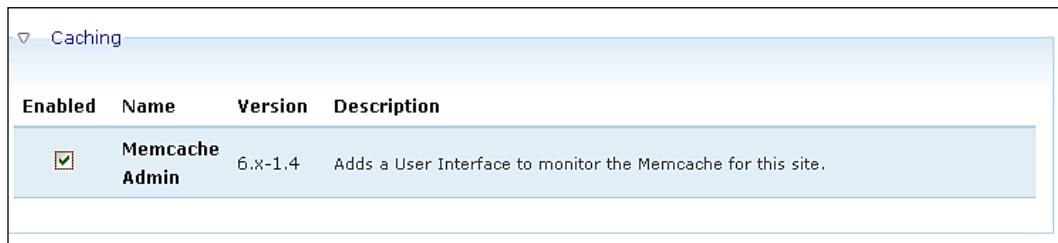
Make sure you comment out the correct line above so that the `memcache.db.inc` is the reference being used.

Also, make sure you're running high enough on your PHP `memory_limit` in your `php.ini`. I had to raise my setting in this site to `96M`.

Bring your site back online and browse to your modules admin list in your Drupal site.

Enabling the Memcache Admin module

Congratulations! You now have the Memcache module running on your development site. Memcache API and Integration provides a sub module called **Memcache Admin** that gives you a simple Drupal-based admin interface to check on your Memcache stats and reporting. The module is located under your Caching modules section. Enable this module and save your module configuration.



The screenshot shows the 'Caching' section of the Drupal Modules administration page. A single module, 'Memcache Admin', is listed and is currently enabled. The table has columns for Enabled, Name, Version, and Description. The 'Enabled' column shows a checked checkbox for 'Memcache Admin'. The 'Name' column shows 'Memcache Admin'. The 'Version' column shows '6.x-1.4'. The 'Description' column shows the text 'Adds a User Interface to monitor the Memcache for this site.'

Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	Memcache Admin	6.x-1.4	Adds a User Interface to monitor the Memcache for this site.

Once enabled, you can browse to this module's admin interface by going to **Reports | Memcache status** through your administrative menu.

Memcache status

The Memcache status page gives you a table showing your site's Memcache status and configuration on one Drupal page. You should see a column listing properties and a column listing values. It should look similar to this:

Memcache status	
127.0.0.1:11211	
Property	Value
pid	312632
uptime	2 hours 40 sec
time	Mon, 11/09/2009 - 13:28
version	1.2.6
pointer_size	32
curr_items	29
total_items	120
bytes	155.76 KB
curr_connections	2
total_connections	34
connection_structures	4
cmd_get	180
cmd_set	120
get_hits	85
get_misses	95
evictions	0
bytes_read	608.52 KB
bytes_written	352.08 KB
limit_maxbytes	64 MB
threads	1
hit_percentage	47.22%
mem_used	0.24%

For example, this page shows you how much memory you're using (0.24%), **uptime** of your site, version of Memcached you are running, **total_items** that you are caching, and the **memory_limit** that Memcache is using here (64MB).

Memcache statistics per page

You can also tell Drupal whether to show Memcache module statistics on each Drupal page when you're logged into the site. To access this setting, go to **Site configuration | Memcache** and check the **Show memcache statistics at the bottom of each page**. As the module notes, these statistics will be visible to any role or user that has permissions to access Memcache statistics.

Home > Administer > Site configuration

Memcache

Show memcache statistics at the bottom of each page
These statistics will be visible to users with the 'access memcache statistics' permission.

Save configuration **Reset to defaults**

memcache_admin module	
access memcache statistics	<input checked="" type="checkbox"/>

Now, browse to one of your nodes and you would see the Memcache statistics shown at the bottom of the page.

```
Memcache statistics
get:
variables
schema
content_type_info
2:235414efa8e61006e64907a167bdde0e
fieldgroup_data
theme_registry:garland
2:131b75eb6b19b2335d70625889990433
links:navigation:page-cid:node/9:1
links:navigation:tree-data:367b971e55cf3486a97ca90fde50b6fe
links:secondary-links:page-cid:node/9:1
```


Viewing the Memcache tables in MySQL

The Memcache API utilizes your default Drupal cache tables in your MySQL database if you are using the `memcache.db.inc` file and configuration. If you view your Memcache module statistics per Drupal page, you'll see that the statistics show the module referencing the MySQL cache tables via the `bins` section. These include `cache`, `cache_filter`, and `cache_menu`. If you view your MySQL database via phpMyAdmin, you'll see the cache tables. This is a nice flexible feature, as this module does not insert any new database tables or special database configuration. Memcache API uses your Drupal default core caching tables to store the cached data.

Running Memcache without saving cache data to your database

Now, let's tweak our `settings.php` file to change the `$conf` array to reference the other include file in our arsenal. This will be the `memcache.inc` file. Comment out the `memcache.db.inc` file, and then remove the comment before your `memcache.inc` file line of code.

Your `settings.php` `$conf` array should read the following:

```
$conf = array(
    // The path to wherever memcache.inc is. The easiest is to simply
    // point it
    // to the copy in your module's directory.
    'cache_inc' => './sites/all/modules/memcache/memcache.inc',
    // or
    // 'cache_inc' => './sites/all/modules/memcache/memcache.db.inc',
);
```

You're now caching your site content without saving to the cache tables in your MySQL database. One thing to be careful about here—as we're not saving our cached data to the MySQL tables, we're relying on the Memcached libraries more here, so make sure you have enough system memory on your server to handle this load. Your website should be responding much faster now, but you also want to make sure you have enough system resources to handle Memcache. The default that Memcache is using is **64MB of system memory**, but this tends to be low. You know that it's running **64MB** by checking your Memcache status in your Drupal site. It tells you this as the value of the `limit_maxbytes` property. You may want to change your Memcached memory to run with **512MB of memory**. To do this, follow these instructions:

1. Go to your `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\memcached Server` in your system registry.

2. Locate the `ImagePath` entry and change it to look like this:
`c:\memcached\memcached.exe -d runservice -m 512`
3. You'll only be able to make these adjustments if you have access to your system registry.

Summary

In this chapter we completed the following steps and now have a good understanding of how Memcached API and Integration module works with our Drupal site:

- Installed the MoWeS Portable suite of applications and Apache server on our Windows XP local machine
- Installed the Memcached service and libraries in order to use the Memcache API and Integration module
- Integrated and tested Memcached with our PHP settings
- Installed and configured the Memcache API and Integration module into our Drupal site
- Enabled the Memcache Admin module in order to display our Memcache statistics on each page load
- Checked our Memcache stats on example nodes

Let's take a break! When we come back in Chapter 8, we'll take a look at contributed modules that are built to work along with the Memcache API module and will help us to enhance our caching mechanisms in our Drupal site even further. This will include the Cache Router, Authenticated User Page Caching, Advanced Cache, and File Based Caching modules. See you back in Chapter 8.

8

Advanced Caching and Contributed Modules for Caching

This chapter will discuss contributed modules for advanced caching including Cache Router, Authenticated User Page Caching, Advanced Cache, **APC (Alternative PHP Cache)**, File Cache, and so on.

In this chapter, we're going to return to our discussion of Drupal caching mechanisms and take a detailed look at a group of contributed caching modules that allow for more granular and advanced cache configurations within our site.

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

- Cache Router
- Authenticated User Page Caching (Authcache)
- Advanced Cache

The focus of this chapter will be how to use contributed caching modules that work and integrate well with Drupal 6.x sites, and how best to implement these modules, first in a development or sandbox type of hosting environment, with the flexibility to implement them later in practice on production websites. We'll see how the modules can contribute to performance optimization and speed on our sandbox and development sites, before rolling them out into a production environment.

Cache Router

The Cache Router contributed module allows you to set up a caching system that assigns specific cache tables to specific Drupal cache technologies. This module works closely with the following cache technologies including Memcache API, the module we worked with in Chapter 7.

- APC
- Database
- eAccelerator
- File
- Memcache API
- XCache

Cache Router is a relatively new and promising module contributed by Steve Rude, available for download at <http://drupal.org/project/cacherouter>. The module allows you to map specific caching technologies or engines to specific cache tables. As mentioned on the module's project page, the engines mainly consist of refactoring previous cache modules such as the Memcache module already discussed.

This module also implements the `page_cache_fastpath` Drupal function for all engines except the database engine, allowing nearly all of Drupal to be bypassed for anonymous users.

Download the latest version of the module `6.x-1.0-rc1` for Drupal 6, extract it to your desktop, and then install it in your `/sites/all/modules` folder. Then enable it from the Drupal modules admin. Also, make sure you have another caching technology, engine, or module installed (such as Memcache API) before using this module.

Enable the module at your modules admin page and save your module configuration.

Caching			
Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	CacheRouter	6.x-1.0-rc1	Controls access to split caching functionality into self contained objects.

Now the default Cache Router settings will work for your site. However, the module project page does point out that there are some tweaks you can make to your `settings.php` file to get the module to work with Memcache as the main caching engine on your site and server.

Open up your `settings.php` file in a text editor and add the following lines of code. We're currently using Memcache API as our caching engine, so I'll add Memcache as the engine. If you were using APC, Database, File, or Xcache, you would add one of those caching engines. Our server will be localhost and then an array of port combinations.

Go ahead and add the following code to your `settings.php` file:

```
$conf['cacherouter'] = array(  
    'default' => array(  
        'engine' => 'memcache',  
        'server' => array('MYIP:MYPORT'),  
        'shared' => TRUE,  
        'prefix' => '',  
        'path' => 'sites/default/files/filecache',  
        'static' => FALSE,  
        'fast_cache' => TRUE,  
    ),  
) ;
```

Now you will have Cache Router working along with your Memcache API caching system.

Cache Router versus Memcache API

There are some drawbacks to using and configuring Cache Router in your site. Memcache API, as we discovered in Chapter 7, provides a nice administrative user interface and shows you statistics on each Drupal page. The Cache Router does not have an administrative interface and will not show you statistics in a visual mode. The project page advertises this with screenshots, but evidently this functionality has been removed from the latest Cache Router version. The module developer is working on integrating stats in future releases of the module as noted in this post on drupal.org: <http://drupal.org/node/338906>.

Authenticated User Page Caching (Authcache)

Let's take a look at another contributed module for caching. This module will allow us to configure caching for our authenticated users. It is frequently referred to as the Authcache module and the project page is here: <http://drupal.org/project/authcache>. The module will cache for both anonymous and authenticated users, and it will increase your page load and Drupal site performance because the page loads will be optimized to about 1-2 milliseconds (according to the module project description). To use this module for anonymous page caching is relatively straightforward. Setting it up for authenticated users will take more detailed and complex crafting.

Authcache will save static HTML versions of Drupal pages and you can configure it to work for specific authenticated user roles in Drupal. The module depends on JavaScript and AJAX to function, and if an anonymous user has JavaScript disabled, they will not see cached versions of the pages. The project page explains how this module works with AJAX in detail.

As with the Memcache API module, you can choose to use Drupal database caching where the module will store all cached pages in the `cache_page` table.

If you do not want to save cached pages to the database (as the Cache Router module works), the Authcache module works with caching engines such as Memcache and the Memcache API module. The advantage – as the project page points out as a general benefit of a caching engine – is that the database will not be storing all the cached data and this will increase your site's performance.

The huge benefit of using this module for your site's performance is that it will greatly increase page load times. According to the module project page, you have a chance of getting page load times under 1 millisecond.

Let's go ahead and try out the module. First, make sure you have a Drupal cache handler module and engine installed, such as Cache Router or Memcache API. We're already using Memcache API, so let's continue to use that. To simplify our caching mechanisms, I'm going to disable Cache Router for the moment and just rely on Memcache API since we have that installed and configured. So, I'll disable Cache Router through my modules admin page and then I'll check my Memcache status to make sure that module is still running without issue.

Download the latest version of the **Authcache** module (`6.x-1.0-rc1`) for Drupal 6.x and install it in your `/sites/all/modules` folder. Enable the module on your modules admin page. Enable both the **Authcache** and the **Authcache Example** modules in your caching section. Save your module configuration.

Caching			
Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	Authcache	6.x-1.0-rc1	Authenticated User Page Caching for logged-in and anonymous users. Required by: Authcache Example (enabled)
<input checked="" type="checkbox"/>	Authcache Example	6.x-1.0-rc1	Example module that displays a cachable Drupal block of customizable user content. Depends on: Authcache (enabled)

Once enabled, the module will give you a message when you reload your modules admin page that you need to update your settings.php file in order to properly configure the Authcache module. It will also give you a link to your **Site Configuration | Performance | Authcache configuration** settings page. Let's look at both of these elements next.

Your settings.php file must be modified to enable Authcache (`$conf['cache_inc']`). See [README.txt](#).

Authcache has been enabled. Please configure your caching settings under [Site Configuration -> Performance -> Authcache](#).

The configuration options have been saved.

Tweaking your settings.php file to support Authcache

Based on the warning message we see once we enable the Authcache module, we need to make some tweaks to our settings.php file to reflect the change to our \$conf array for our cache_inc. As you recall we're currently using the Memcache API module and referencing this in our \$conf array. We need to tweak this and reference our new Authcache module, and also add a reference to our Memcache so that we can use both of these integrated. Open your settings.php file and comment out the previous array you configured for Memcache. Then add the following lines of code to your file. First, to reference the Memcache API module, we're going to add this:

```
$conf ['memcache_servers'] = array('localhost:11211' => 'default');
```

Then to reference our Authcache module we'll add this:

```
$conf ['cache_inc'] = './sites/all/modules/authcache/authcache.inc';
```

Once you make these tweaks save your settings.php file and then refresh your main modules admin page. The warning message should disappear.

Remember that if no caching engine or module such as Memcache API is installed, the Authcache module will use the default Drupal core database cache tables to work.

Configuring the Authcache module

To configure the Authcache module settings, go to **Site Configuration | Performance | Authcache** or by visiting this URL: <http://localhost/drupal/admin/settings/performance/authcache>. You should see a form that looks similar to this:

The screenshot shows the 'Authcache Configuration' page under 'Performance'. The 'Authcache' tab is selected. The 'Configuration' tab is active. A note at the top says: 'Note: Page compression is not enabled! It is strongly recommend that you turn this setting on through Drupal to lower your cache memory and allow for faster page response times.' Below this, the 'Authcache Roles' section is expanded, showing checkboxes for 'anonymous user', 'authenticated user (without additional roles)', 'content editors', and 'site admin'. A note below these checkboxes states: 'If no roles are selected, Authcache page caching will not be enabled. Unchecked roles and the admin account (uid #1) will never be cached.' There is also an unchecked checkbox for 'Invalidate all user sessions'. A note next to it says: 'This is required when you first enable the Authcache module & anonymous caching, otherwise logged-in users may receive pages intended for anonymous visitors. All users will need to relogin after this.' At the bottom of the page is a 'Save & clear cached pages' button.

This settings page shows us that we can set caching for specific roles. You can set caching for all of your site's roles or just for specific roles such as anonymous users. The settings also tell you here that the super user admin role account will never be cached. Let's check the **Enable caching for specified user roles** boxes next to the **anonymous user** and the **authenticated user**.

Check the box next to **Invalidate all user sessions**. By doing this your authenticated users will be forced to login to the site again (if they are currently logged in for their user session).

Expand the **Authcache Debugging/Development** section and check the boxes for **Enable debug mode for all roles** and **Enable for session if http://localhost/drupal/authcache_debug is visited**. Also, check the box next to **Benchmark database queries** so that you can gather statistics on the number of queries used, query time, and percentages related to the page load time.

Click on the **Save & clear cached pages** button.

Authcache Configuration

Drupal core **Authcache**

Configuration Page Caching Settings

Note: Page compression is not enabled! It is strongly recommend that you turn this setting on through Drupal to lower your cache memory and allow for faster page response times.

▼ **Authcache Roles**

Enable caching for specified user roles:

- anonymous user
- authenticated user (without additional roles)
- content editors
- site admin

If no roles are selected, Authcache page caching will not be enabled. Unchecked roles and the admin account (uid #1) will never be cached.

Invalidate all user sessions

This is required when you first enable the Authcache module & anonymous caching, otherwise logged-in users may receive pages intended for anonymous visitors. All users will need to relogin after this.

▼ **Authcache Debugging/Development**

Debug mode will display the page's cache statistics, benchmarks, and Ajax calls.

- Enable debug mode for all roles.**
- Enable for session if http://localhost/drupal/authcache_debug is visited.**

Enable for specified users:

Enter a comma-delimited list of usernames to enable debug mode for. Users will need to relogin to enable debug mode.

- Benchmark database queries**

This will display the number of queries used, query time, and the percentage related to the page's total render time.

Save & clear cached pages

Once saved, Authcache will show you a message stating the following:

- 0 user sessions have been invalidated
- Drupal's built-in page caching for anonymous users has been disabled to avoid redundant caching
- Your Authcache settings have been saved
- Cached pages have been cleared

You may also see a message telling you to enable Page compression in your performance settings. This will help to optimize your page load performance even more. You can decide whether you want to enable Page compression or not.

Page Caching Settings

Click on the **Page Caching Settings** tab on your **Authcache Page Caching Settings** admin page. This will launch the **Caching Ruleset** configuration. This page and form is only visible once you set up the initial Authcache configuration as per the instructions in the previous section. You should see a screen that looks like the following:

The screenshot shows the 'Authcache Page Caching Settings' administrative page. The top navigation bar includes tabs for 'Drupal core' and 'Authcache'. Below the navigation, there are two tabs: 'Configuration' and 'Page Caching Settings', with 'Page Caching Settings' being the active tab. A collapsible section titled 'Caching Ruleset #1' is expanded, showing the configuration for this ruleset. Under 'Cache specified pages:', the 'Cache every page except the listed pages.' option is selected. The 'Pages:' input field contains a list of Drupal paths: 'admin*', 'user*', 'node/add/*', 'node/*/edit', 'node/*/track', and 'tracker*'. A note below the input field explains that paths starting with a '/' are considered wildcards. A link provides instructions for using PHP code in the input field. Under 'Apply to these roles:', two checkboxes are checked: 'anonymous user' and 'authenticated user (without additional roles)'. At the bottom of the ruleset configuration, there is a button labeled 'Add new ruleset...' with a plus sign icon. Another collapsed section at the bottom is titled 'Non-HTML Cached Pages'.

This form allows you to cache specific pages on your site using Authcache or to cache all pages. The Cache-specified pages section is set up similar to how Boost works, as well as the block administration page. You'll notice here that Authcache adds some default pages that are not going to be cached including any /admin page. The full list of these pages looks like this:

```
admin*
user*
node/add/*
node/*/edit
node/*/track
tracker*
comment/edit*
civicrm*
cart*
*/ajax/*
*/autocomplete/*
ajax_comments*
```

Let's leave these sensible Authcache defaults in place. You can also choose which roles to apply this specific ruleset. If you want to add different rulesets for each role, you can then click on the **Add new ruleset...** button and this will add an entirely new ruleset that you can configure for your role. You can see that this module provides a lot of flexibility and functionality similar to the Boost module.

Clicking on the **Add new ruleset...** button, you'll then be presented with a new ruleset form:

Caching Ruleset#2

Cache specified pages:

- 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:

To delete this ruleset, leave the textbox empty. Enter one page per line as Drupal paths. The '*' character is a wildcard. Example paths are 'blog' for the blog page and 'blog/*' for every personal blog. <front> is the front page. If the PHP-mode is chosen, enter PHP code between <?php ?>. Note that executing incorrect PHP-code can severely break your Drupal site.

The ID for excluding or including this element is: edit-ahah-1-fieldset-pages - the path is: authcache/ahah

Apply to these roles:

- anonymous user
- authenticated user (without additional roles)

Add new ruleset...

When you are satisfied with your configuration, click on the **Save & clear cached pages** button again.

Testing the Authcache module and its caching mechanism

Let's test out the module now to make sure it's caching our pages for anonymous and authenticated users.

There are two methods of testing whether the module is working correctly. Let's look at both.

Checking the AuthcacheFooter code

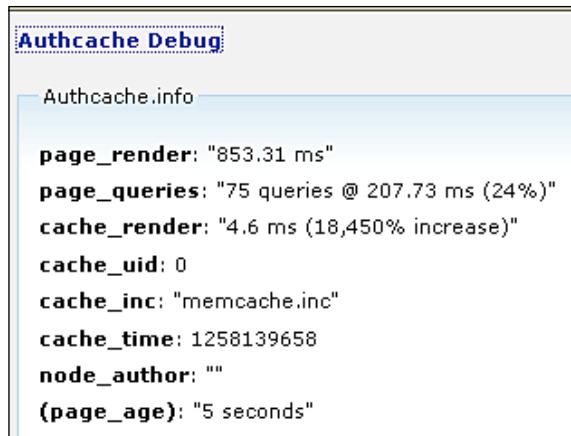
When the module is enabled and configured, logout of the website and view one of your nodes as an anonymous user. If you view source on the node, you should see the following script code and JSON object at the bottom of your HTML code. It should look like this and it will be commented out as the Authcache Footer JSON object:

```
<!-- Authcache Footer JSON -->
<script language="JavaScript">
var authcacheFooter = { "info": { "page_render": 853.31, "page_
queries": "75 queries @ 207.73 ms (24%)", "cache_render": "-1",
"cache_uid": 0, "cache_inc": "memcache.inc", "cache_time": 1258139658,
"node_author": "" }, "ajax": { "q": "node/8" } };
</script>
```

This code actually gives you some debug information including `page_queries`, `cache_render`, and `memcache cache time`, all for that specific node.

Checking the Authcache Debug window

You can also enable and give permissions to specific roles to view the debug statistics for a node via the **Authcache Debug** window. This is a JavaScript-powered window that appears in the upper-left corner of your site via the **Authcache Debug** link. If you click on this link, you'll see the debug stats expand. You should see something similar to this:



The screenshot shows a 'Authcache Debug' page with a single section titled 'Authcache.info'. It contains the following data:

page_render:	"853.31 ms"
page_queries:	"75 queries @ 207.73 ms (24%)"
cache_render:	"4.6 ms (18,450% increase)"
cache_uid:	0
cache_inc:	"memcache.inc"
cache_time:	1258139658
node_author:	""
(page_age):	"5 seconds"

If you only want specific roles to be able to view this debug information, tweak the permissions for viewing it. Uncheck the **Enable debug mode for all roles** if you do not want your anonymous users viewing your debug window and statistics.

Remember that you also enabled caching for your authenticated user role. Log in as an authenticated user to test this out. Once logged in, view a node and you should see the debug information. This shows you that caching is working for authenticated users as well.

You now have a clear overview of the Authcache module, have put it into action on your site, and used it for both anonymous and authenticated user page caching. Let's move on and look at another module that we can use for caching.

Advanced cache

Now, what if you like to take advantage of several new cache opportunities, such as setting up caching for your site's comments, forums, search, taxonomy, and more? This is where the Advanced cache module will help out. This module provides advanced level caching by way of running a set of patches to your site's code that will allow for caching in areas of Drupal that do not currently cache by default. This includes comments, taxonomy terms, trees, vocabs and terms-per-node, path aliases, and search results.

To use this module, you need to install the Advanced cache module that currently has a production version for Drupal 5.x and development versions for Drupal 6.x. You will also need to apply some or all of the patches that come with the module. You can apply patches through command line or shell if you have access to your server via shell. The main Drupal module project page, <http://drupal.org/project/advcache>, contains detailed descriptions of all the patches that ship with the module. Of course, as with any dev version module, there are noted bugs in some of these patches, so you use this module and its patches at your own risk.

Not only do these patches allow for modules, such as Memcache, to do an even greater job, they will increase performance for authenticated users. Once installed from <http://drupal.org/project/advcache>, we may now patch all using `all_patches.patch` or choose the select few that we are interested in.

We will now touch on what additional functionality each of the patches provides, helping us decide which patch we should apply.

block_cache.patch

Caches enable blocks per user and then per theme for both authenticated and anonymous users. This patch is compatible with the blockcache module. The module project page claims that this module will take 1.5 milliseconds off of your page load times.

comment_cache.patch

Caches comment markup. It will still display changes made via settings or new comments (including the new indication).

forum_cache.patch

This patch caches the forum structure on your site, but beware that this patch is buggy and broken, and you are not advised to use it.

node_cache.patch

Caches node displays for users. When your site is made up of nearly all anonymous users, this is a patch you may wish to skip. However, if many users authenticate to your site and do not contain multiple roles, this patch will be very beneficial.

path_cache.patch

This patch will cache path aliases. So if you are using the path module, you most likely would like to apply this patch, shaving off a single query per internal link displayed on the site (not including those previously cached in blocks, nodes, and so on). This patch is also buggy and broken at this point, so beware and use with caution.

search_cache.patch

Similar to node_cache.patch, this will cache search results per authenticated user, per role. This patch is also broken at this point.

taxonomy_cache.patch

Caches taxonomy trees, terms, and vocabularies.

Let's go ahead and install the module and apply one of these patches. First, download the development version of the Advanced Cache module for Drupal 6.x. The current version is 6.x-1.x-dev. Download and install in your /sites/all/modules directory.

Enable the module on your admin modules page.



A screenshot of the Drupal administration interface showing the 'modules' page. The 'Caching' module is selected. A table lists the module details:

Enabled	Name	Version	Description
<input checked="" type="checkbox"/>	Advanced cache	6.x-1.x-dev	Introduces advanced caching techniques to Drupal core

If you try enabling the dev version of the module, you may receive the following parse error:

```
Fatal error: Cannot redeclare advcache_update_1() (previously declared in
C:\www\drupal\sites\all\modules\advcache\advcache.install:58) in
C:\www\drupal\sites\all\modules\advcache\advcache.install on line 238
```

To fix this, you will need to apply the following patch as per this troubleshooting document on drupal.org (<http://drupal.org/node/430700>).

Here's the patch:

```
--- advcache.install.bak 2009-04-11 11:00:19.000000000 +0200
+++ advcache.install 2009-04-11 11:02:22.000000000 +0200
@@ -206,33 +206,3 @@
    }
}

-
-function advcache_update_1() {
-  switch ($GLOBALS['db_type']) {
-    case 'mysql':
-    case 'mysqli':
-      $ret[] = update_sql("CREATE TABLE {cache_forum} (
-        cid varchar(255) NOT NULL default '',
-        data longblob,
-        expire int NOT NULL default '0',
-        created int NOT NULL default '0',
-        headers text,
-        PRIMARY KEY (cid),
-        INDEX expire (expire)
-      ) /*!40100 DEFAULT CHARACTER SET UTF8 */ ");
-      break;
-
-    case 'pgsql':
-      $ret[] = update_sql("CREATE TABLE {cache_forum} (
-        cid varchar(255) NOT NULL default '',
-        data bytea,
-        expire int NOT NULL default '0',
-        created int NOT NULL default '0',
-        headers text,
-        PRIMARY KEY (cid)
-      )");
-      db_query("CREATE INDEX {cache_forum}_expire_idx ON {cache_
forum} (expire)");
-      break;
-    }
-  return $ret;
-}
```

Install the patch, and then try refreshing the browser to see if the parse error disappears. The patch basically tells you to delete the entire second occurrence of this set of code because it's duplicated in the `advcache.install` file. Remove the second instance of this function.

The development version of this module for Drupal 6.x currently only has the following patches for you to try:

- node.patch
- nath.patch
- taxonomy.patch

To run a patch, open up your command line or command shell utility if using Windows and type in the following:

```
Cd C:\www\drupal patch < sites/all/modules/advcache/
DRUPAL-6-10/taxonomy.patch
```

Alternatively copy the patch file in question to the root of your /advcache folder and then run the patch from that directory.

APC (Alternative PHP cache)

At the time of writing, the APC module has been revoked for Drupal 5 and 6.x. This is due to the fact that APC will be included in Drupal 7 with the Drupal 7 caching system. The Drupal 7.x caching system will allow for custom cache backend configuration, thus making the module unnecessary and not supportable in earlier versions of Drupal. The module maintainer informs the developer that they can use other caching modules, such as Cache Router, to perform similar caching operations. See the APC module page here: <http://drupal.org/project/apc>.

File Cache module

The File Cache module is currently available for Drupal 5.x, but does not have a version for Drupal 6.x. The module maintainer recommends using the Memcache API for any type of complex file caching that you may need to implement. The project page for this module is here: <http://drupal.org/project/filecache>

Summary

In this chapter, we completed the following steps and now have a good understanding of how more complex and advanced caching modules work with our Drupal site. We have:

- Installed and configured the Cache Router module
- Installed and configured the **Authcache module**
- Tweaked our `settings.php` file to support the Authcache module
- Integrated and tested Authcache
- Installed and configured the Advanced Cache module
- Briefly glanced at the Alternative PHP cache module (APC)
- Briefly discussed File caching

Let's take a break! When we come back in Chapter 9, we'll take a look at how best to run a high performance Drupal multisite. See you back in Chapter 9.

9

Multisite Configuration and Performance

In this final chapter, we are going to set up and configure a Drupal multisite environment on our localhost development server. The benefits to running a multisite installation include allowing multisites to share one Drupal database and core module configuration, or to share a core module configuration and allow each site to have its own database. We will also look at how multisite installations can help to boost performance for your overall Drupal configuration and help to increase performance on your server whether it be a shared or dedicated server environment.

By the end of this chapter you will know how to install and configure a Drupal multisite including the following configuration steps:

- Creating multisite folders and `settings.php` files
- Allowing each multisite to talk to its own MySQL database
- Configuring your HTTPD configuration file (`httpd.conf`) to support Virtual Hosts
- Tweaking the Windows system driver for Hosts
- Using caching in a multisite environment
- Looking at Shared database tables versus individual databases
- Installing modules and themes to multisites
- Using `update.php` on multisites
- Exploring Resources for Drupal multisite development

Let's go ahead and set up a Drupal multisite environment.

Using Drupal multisite

What is Drupal multisite and why should we use it? Multisite allows you to run multiple Drupal websites and instances from one main Drupal core codebase. As we have seen throughout this book, when we install and configure Drupal, we are using Drupal's main core system and database to power and drive our website. This includes the core /modules and /themes folders, as well as /scripts, /profiles, /misc, and /includes. **The flexibility of Drupal allows us to install additional websites** (as many as we need) and share all of our core files, modules, and themes within these multisites. This helps tremendously when we are planning to build a large website environment for a company or organization that has multiple departments, divisions, or stakeholders. We can offer these types of clients a large scale Drupal multisite. The sites will be more flexible, will perform better, and be easier to maintain.

One benefit here is that when you **need to upgrade or patch a core or contributed module or theme** and you're sharing those modules and themes with the other Drupal sites in your /sites folder, you only need to patch once and the updates will work across all sites. You can easily see the benefits from a maintenance and performance perspective here. If you need to update the modules only once and the updates are applied to all your sites, then this will save you time from a maintenance perspective. It will also provide for a much more consistent development and performance framework because all the sites will be running smoothly from the same codebase.

The other large benefit of a multisite is that you can install contributed and custom themes and modules to specific sites. All of your multisites can be running different custom themes and they can run different contributed modules. They will also have their own file system for associated images, documents, and other files. Finally, you can tweak performance per site so that if you want to run caching on one site, you can, without that caching mechanism affecting your other multisites.

We're going to focus on running multiple Drupal sites that all work with their own MySQL database. The benefit to this is that you have data separation for your contributed content and module code. Each site has its own database and therefore if issues arise with one site, these issues will not necessarily affect the other sites in your multisite environment. This type of environment will also help you to troubleshoot better. You can also run Drupal multisite from one shared database and we'll discuss this method as well. Both methods have their advantages and disadvantages.

Configuring multisite in a localhost environment

We are going to configure our Drupal multisite environment on our localhost development server, either on Windows or Linux. I'm going to walk through the steps of setting this up on Windows because there's an additional trick for Windows users to make sure they can get a Drupal multisite environment functioning easily in a Windows development environment. However, these instructions will also work on a Linux system or a MAMP system on Mac OS. We'll also briefly discuss how to best approach setting up multisite environments on production servers including Ubuntu Linux, and using a more common cPanel solution on shared servers. First let's start by configuring a multisite development environment on our local development environment.

You have already prepped your multisite environment in the previous chapter when we set up a Drupal site on our localhost using MoWeS Portable applications. When doing this we installed the Apache web server, PHP, and MySQL on our local development environment. We now have Apache running and we have access to phpMyAdmin to do our database work by going here: <http://127.0.0.1/phpmyadmin/>.

We also installed a Drupal website in our C:\www root web docs folder. Our site is simply called /drupal and is located here via the web browser: <http://localhost/>.

So, we have the base Drupal core site set up. Our core Drupal site is working smoothly, and in the previous chapter recall that we also configured Memcache API and Authcache to run caching mechanisms on our site. The next thing we're going to do is configure this core /drupal site to be the core codebase for two new Drupal sites that we're going to install and configure. To make things simple, I'm going to just duplicate this site and use the same content for the two new sites. I'll do this by exporting the data from the core /drupal site's database and importing that data into two new MySQL databases that we're going to create. Once we complete the multisite configuration, all three of our Drupal sites will be running from their own database, but they will be sharing the core codebase, core modules, and themes.

Creating the multisite folders

The first thing we need to do is create folders for each site we want to run. To do this, go to your main /drupal site folder and open the main /sites folder. For Drupal multisite, you need to add your site folders into the main core /sites folder. So, we're going to add two sites and create the following folders in our /drupal/sites folder:

- site1
- site2

You should now have your folder directory looking like this:

```
/sites/site1  
    /site2
```

The next step is to copy your main /sites/default/settings.php file and paste a copy of it into each of your multisite folders. So you'll have:

```
/site1/settings.php  
/site2/settings.php
```

The reason you do this is that each multisite needs to have its own settings.php configuration file so that you can tell that site what database to work with and also what base_url to use. We'll change those settings in a moment.

You must also create /files folders inside each of your site's directories. So you'll have:

```
/site1/files  
/site2/files
```

Setting up databases for your multisite

Now we already have a core database for our /drupal folder. We're going to create two new databases (one for each of our new sites). To do this I'm going to use the same data that my core database contains. So the first thing I'm going to do is export a clean database dump of the /drupal database and save that locally on my desktop. Then I'll import that data into each new database that I configure. Here are the steps:

1. Fire up your phpMyAdmin at: <http://localhost/phpmyadmin>.
2. Locate your main site's database and open it.

- Click on the **Export** tab and then select **Save as file** and click on the **GO** button.

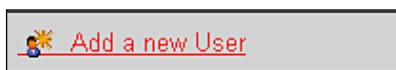
The screenshot shows the 'Export' tab selected in the top navigation bar. The main area is titled 'View dump (schema) of database'. On the left, a list of tables is shown with checkboxes for selecting them. Below this is a list of export formats (CSV, CSV for MS Excel, Microsoft Excel 2000, Microsoft Word 2000, LaTeX, Open Document Spreadsheet, Open Document Text, PDF, SQL, XML, YAML), with 'SQL' selected. To the right are several configuration sections:

- Options**: Includes checkboxes for adding a custom comment, enclosing the export in a transaction, disabling foreign key checks, and setting SQL compatibility mode (set to 'NONE').
- Structure**: Includes checkboxes for adding DROP TABLE / VIEW / PROCEDURE / FUNCTION, IF NOT EXISTS, AUTO_INCREMENT value, enclosing table and field names with backquotes, and CREATE PROCEDURE / FUNCTION.
- Add into comments**: Includes checkboxes for creation/update/check dates, relations, and MIME type.
- Data**: Includes checkboxes for complete and extended inserts, and options for delayed, ignore, or hexdecimal inserts. It also specifies a maximal length of 50000 and an export type of 'INSERT'.
- Save as file**: Contains fields for a file name template ('__DB__') and compression options ('None', 'zipped', 'gzipped'), with a 'remember template' checkbox checked. A 'Go' button is at the bottom.

- This will save a .sql file to your desktop.
- Now load your main phpMyAdmin screen and click on the **Privileges** link.

[Privileges](#)

- Click on **Add a new User**.



7. Type in a new username in the text field box and set the host to localhost. Type and confirm a password into the **Password** field. This will be the username and password for your new site's database, so make a note of this because you will be re-entering this information into your site's `settings.php` file. You will need to know the database username, password, and database name. A hint for what to name your new site's databases: call them `_site1` and `_site2`, using your main core's database name as the appended name. So, for example, my new site's databases are called `drupal6_site1` and `drupal6_site2`.
8. Check the radio button next to **Create database with same name and grant all privileges**.
9. Click on the **Check All** link in the **Global privileges** section and then click on **Go**. This will create the new database.

Add a new User

Login Information

User name:	Use text field:	drupal6_site1
Host:	Local	localhost
Password:	Use text field:	*****
Re-type:	*****	
Generate Password:	Generate	Copy

Database for user

None
 Create database with same name and grant all privileges
 Grant all privileges on wildcard name (username\%)

Global privileges (Check All / Uncheck All)

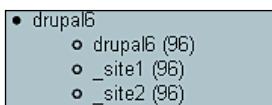
Note: MySQL privilege names are expressed in English

Data	Structure	Administration	Resource limits
<input checked="" type="checkbox"/> SELECT <input checked="" type="checkbox"/> INSERT <input checked="" type="checkbox"/> UPDATE <input checked="" type="checkbox"/> DELETE <input checked="" type="checkbox"/> FILE	<input checked="" type="checkbox"/> CREATE <input checked="" type="checkbox"/> ALTER <input checked="" type="checkbox"/> INDEX <input checked="" type="checkbox"/> DROP <input checked="" type="checkbox"/> CREATE TEMPORARY TABLES <input checked="" type="checkbox"/> CREATE VIEW <input checked="" type="checkbox"/> SHOW VIEW <input checked="" type="checkbox"/> CREATE ROUTINE <input checked="" type="checkbox"/> ALTER ROUTINE <input checked="" type="checkbox"/> EXECUTE	<input checked="" type="checkbox"/> GRANT <input checked="" type="checkbox"/> SUPER <input checked="" type="checkbox"/> PROCESS <input checked="" type="checkbox"/> RELOAD <input checked="" type="checkbox"/> SHUTDOWN <input checked="" type="checkbox"/> SHOW DATABASES <input checked="" type="checkbox"/> LOCK TABLES <input checked="" type="checkbox"/> REFERENCES <input checked="" type="checkbox"/> REPLICATION CLIENT <input checked="" type="checkbox"/> REPLICATION SLAVE <input checked="" type="checkbox"/> CREATE USER	<small>Note: Setting these options to 0 (zero) removes the limit.</small> MAX QUERIES PER HOUR: 0 MAX UPDATES PER HOUR: 0 MAX CONNECTIONS PER HOUR: 0 MAX USER CONNECTIONS: 0

Go

10. Repeat steps 5 to 9 to create a database for your second multisite site. Now you have both databases created. The username for each database is the same as the database name, and the password used for each is `drupal`. You will want to use a more complex password combination on a production site or server, but here we can keep things simple. Make a note of these details, as you'll need them when we edit the `settings.php` file for each new site.

11. Now you need to import the exported data (from your main core database) into these new databases.
12. Click on the database link in the left sidebar of phpMyAdmin and then click on the **Import** tab.
13. Browse for your saved .sql file and then make sure the format of the file to be imported is set to **SQL**.
14. Click on **Go** and this will import the database tables to your new database.
15. Repeat steps 11 to 14 for your second site's database.
16. You should now have a set of three databases with the same data in your phpMyAdmin. The main **drupal6** database is your core site's database. The _site1 and _site2 databases are for each of your site's multisite installations. You should see something in phpMyAdmin that looks like the following screenshot:



Tweaking settings.php for each site

Now that you have duplicated your database and set up your databases for your new multisites, let's make tweaks to the `settings.php` file for each of these new sites.

Open up your `/site1` folder and open the `settings.php` file to edit it in a text editor. You will need to change the permissions on the `settings.php` file before editing it because the file is normally set to read-only. To do that in a Windows environment, right-click on the file and select **Properties** from your menu options. Then on your properties **General** tab, uncheck **Read-only** in the **Attributes** section. You can also click on the **Security** tab in your **Properties** box and then check the permissions for your system **Users**. Make sure that system **Users** (or at least **Administrators**) if you are currently logged into your computer as its main admin user) have write permissions to the file. You can also change these permissions using FTP or a Filemanager utility. With FTP on your remote host side of the FTP client, right-click on the `settings.php` file and you should be able to set the permissions for the file. In a Filemanager utility via cPanel, you should see a button (**Change Permissions**) to change permissions for the file.

Download at [Wow! eBook](#)

We need to update the \$db_url to reference our new _site1 database. You insert your database user, password, and then the database name in this line of code in the settings.php file. This site will have its own database, but will share core resources with the main Drupal site configuration, as those database tables have been copied over into the _site1 database. Change your db_url to:

```
$db_url = 'mysqli://drupal6_site1:drupal@localhost/drupal6_site1';
$db_prefix = '';
```

Do the same for the settings.php file of /site2. That site's \$db_url should read:

```
$db_url = 'mysqli://drupal6_site2:drupal@localhost/drupal6_site2';
$db_prefix = '';
```

If you are installing multisites, but sharing the additional sites with the main core Drupal site database, you'll need to reference the same \$db_url, but update your \$db_prefix to note the additional site name. This way you can share the same database and make sure each site has its own database table prefix. That code would look like the following for site1:

```
$db_url = 'mysqli://drupal6:drupal@localhost/drupal6';
$db_prefix = 'site1_';
```

This would be the site2 configuration:

```
$db_url = 'mysqli://drupal6:drupal@localhost/drupal6';
$db_prefix = 'site2_';
```

For now let's keep this configuration using different databases as per my instructions above.

Editing your Apache configuration

The next step in multisite configuration is to add virtual host entries to our Apache web server configuration. You need to tell Apache to look for these additional sites in our multisite configuration when a site visitor loads the site's web address in the web browser. To do this we use a virtual host entry in the Apache configuration file.

Locate your Apache httpd.conf file. It should be in the following folder location on your localhost server:

```
C:\apache2\conf\httpd.conf
```

Open this .conf file in your text editor. You will need to know what port your localhost is listening on. Usually it's port 80, but if you are not sure check your server configuration. The default settings in the httpd.conf file should be correct.

First look for the following code in your `httpd.conf` file and make sure it's uncommented. This code should be in Section 3: Virtual Hosts of your conf file. We want the following uncommented:

```
NameVirtualHost *:80
```

Below this you will see some `VirtualHost` examples. You can use the examples the configuration file provides, and just make sure they are uncommented and have the correct settings as per your configuration. This is what the code should look like:

```
<VirtualHost *:80>
    DocumentRoot /www/drupal
    ServerName localhost
</VirtualHost>

<VirtualHost 127.0.0.1:80>
    ServerName site1
    DocumentRoot "C:/www/drupal"
    <Directory />
        Options All
        AllowOverride All
        #Simply allow all directives to be overridden in .htaccess
        Order allow,deny
        Allow from localhost
        #Only allow access from localhost
    </Directory>
</VirtualHost>

<VirtualHost 127.0.0.1:80>
    ServerName site2
    DocumentRoot "C:/www/drupal"
    <Directory />
        Options All
        AllowOverride All
        #Simply allow all directives to be overridden in .htaccess
        Order allow,deny
        Allow from localhost
        #Only allow access from localhost
    </Directory>
</VirtualHost>
```

Make these code additions and tweaks, then save your `httpd.conf` file and close it. Then stop your MoWeS Apache web server and restart it so that the above changes take effect.

Here's what the above code does. The first section of code tells Apache to use localhost as the server and to set the document root for that server to the Drupal core site you have configured. In this case, the document root is /www/drupal. This is reflected in this code excerpt:

```
<VirtualHost *:80>
    DocumentRoot /www/drupal
    ServerName localhost
</VirtualHost>
```

The next set of code is the `VirtualHost` directives to tell Apache to look for your multisites. First, we have a selection that names `ServerName` as the multisite entry (`site1` or `site2`) and then instructs it to use the same document root as the main core site /www/drupal. This allows Apache to look for these multisites in your main core /sites folder and then to deliver them to the site visitor when they type in `site1` or `site2` in the web browser URL menu.

Once you restart the web server, you will need to make one final tweak on Windows computers to make sure this new multisite configuration works for you.

Tweaking the hosts driver file on Windows

Now, in order for this to work on Windows, we need to tweak the driver file for Host configurations in the system settings folder. Browse to the `C:\WINDOWS\system32\drivers\etc` folder and look for the hosts file. If you do not see the file, then it's most likely hidden. To show the file on your Windows system, go to the **Tools** menu at the top of your folder window and select **Folder Options**. Then click on the **View** tab and select the **Show hidden files and folders** radio button that resides in the **Hidden files and folders** folder icon. Open that file in the text editor.

Look for the hostname code at the bottom of the file and make sure it lists localhost as well as your multisites. You should add the following if it's not there already:

```
127.0.0.1      localhost
127.0.0.1      site1
127.0.0.1      site2
```

Make sure this code is uncommented. Save this file.

Tweaking the Base URL

You should also make sure to tweak your Base URL setting in each of your site's settings.php files. Open each settings.php file and look for the Base URL section. You will need to uncomment the line of code that reads:

```
$base_url = 'http://localhost'; // NO trailing slash!
```

Make sure this has the URL address for your site. So the above is for the main localhost site and the following are for your multisites (in their respective settings.php files):

```
$base_url = 'http://site1'; // NO trailing slash!  
$base_url = 'http://site2'; // NO trailing slash!
```

Loading your multisites

That's it! You are ready to fire up all of your sites in your web browser. First, load the main localhost site, which is your main core Drupal site. Load that by going to: <http://localhost/>.

Now load site1, which will be: <http://site1/>.

Site2 will be: <http://site2/>.

Each of these sites is now behaving as their own Drupal site with their own databases. We will now work with each site to show you how you can install and enable modules independently on each site or share core and contributed modules from site to site. Congratulations on getting multisite configured and working!

Testing your multisite configuration

Now that we have Drupal multisite working in our localhost environment, let's go ahead and test it to make sure all three databases and sites are working correctly and independently. First, go to your site2 site and login to it here: <http://site2/user>. Your username and password on this site will be the same as on your core site, as you simply duplicated the same database to use with this new multisite. Go ahead and login.

Now, if you have caching enabled on this site, let's disable the caching modules for the moment so that we can immediately see our content edits occur for anonymous users. Disable the Authcache and Memcache API modules if you have enabled them in site2, and also clear this site's database performance cache.

Now open one of your site's nodes and make some edits to the node. For example, I'm going to edit the home page pane content that states **Welcome to Everything Drupal!** and I'm going to add a note in that pane that specifies that this site is Drupal Web site Multisite 2!. Here's a screenshot of what that content looks like:

Welcome to Everything Drupal!

Welcome to the Everything Drupal Web site Multisite 2! You can learn all about Drupal on this site.

Now, logout of `site2` and browse back to your main localhost core site here: `http://localhost/`. Here you should not see the above text tweaks you made to the content on `site2`. Here it should look like your original, or:

Welcome to Everything Drupal!

Welcome to the Everything Drupal Web site! You can learn all about Drupal on this site.

This shows that content editing is working correctly in your multisite. The `site2` edits are posting to the `site2` database, but not to your core site's database.

Using core and contributed modules in multisite

One large benefit of using Drupal multisite is that you can install all of your core and contributed modules into the main core Drupal site installation and then enable or disable modules in each of your sites in your `/sites` folder. So your main site can run specific core and contributed modules while your multisites can run their own choice of core and contributed modules. Let's look at this in detail to see how it works. First, login to your core Drupal site and browse to the modules admin page to view the modules you have currently installed.

Notice that we have the Webform and Panels contributed modules installed on our core site. Let's leave all of these modules enabled on our core site. Now, login to one of your new multisite installs, such as `site1`, and browse to the modules admin page.

On `site1`, uncheck the boxes next to Webform and Panels to disable these modules, only on our `site1` site. Save the configuration on your modules page. This will disable these modules on `site1`. But if you browse back to your main localhost site, you will see that these contributed modules are still enabled. You will see the power and flexibility of Drupal multisite with this example.

Installing modules and themes to a multisite

Another flexible bonus to a multisite installation is the ability to install and enable contributed modules on your multisites, but not on your main core site. Here you just need to remember that when the contributed module receives a security upgrade or patch, you'll need to update the module in the `/sites/site/modules` folder for that specific site and then run `update.php` on that site so that the site updates the database schema correctly.

Let's go ahead and install and enable a module in our `site1` site. Download a module such as the Drupal **AddThis** Button module and extract this module in your `/sites/site1/modules` directory. If you do not have a `/modules` or `/themes` directory in `/sites/site1/`, make sure to create these folders. These will contain just the contributed modules and themes for your `/site1`.

Once extracted, browse to your `site1` modules page, you should see the **AddThis** module in your list. However, if you browse to your modules admin page in your localhost or your `site2` sites, you will not see the module in the list.

Other			
Enabled	Name	Version	Description
<input type="checkbox"/>	AddThis	6.x-2.8	Creates AddThis button as a block, to be used in themes and to node links.

Once enabled, this **AddThis** module will show up in your `site1` administration menu. Look for **Site configuration | AddThis**. Browse to the **AddThis** configuration page. If you try this in `site2` or localhost, you will see that you cannot get to this configuration page, as the module is not enabled in those sites.

Setting themes per multisite

With Drupal multisite you can also set different themes as the default enabled theme on each multisite. So, we can run one specific theme, such as Garland, on our main localhost and a different core or contributed module on our `site1` and/or `site2`. Let's go ahead and try this. Leave the core Garland theme set as the default enabled theme on your localhost site. Then browse to your themes admin page in either `site1` or `site2` and enable a different core theme as your default enabled theme.

I'm going to set the `site1` theme to Bluemarine. I could also choose a contributed or custom theme if I have one installed in my `/site1/themes` folder. Save your theme configuration. You will also notice that if you have a contributed theme installed in your core site's `/sites/themes` folder, then these contributed themes will show up in your multisites' themes admin page and you can choose to enable those as well.

Now if you browse to your `site1` home page, you should see that Bluemarine is active as the default site theme. If you browse back to your localhost or `site2` sites, you will see that the Garland theme is still active and enabled.

Caching and multisite

As you work through these examples, you may notice that changes do not take effect immediately on each site or that if you make a change on `site1` or `site2`, then those changes also take effect on localhost. This is not supposed to happen. But if you have caching enabled from your previous single site experiments using Memcache API, Memcached, or Authcache, you will need to disable those modules and also comment out your caching code from your `settings.php` files.

You'll recall that the configuration instructions for enabling caching in a multisite environment are different from a single site environment. So, let's first disable the Memcache and Authcache code, and then come back to this to tweak later if we want to enable caching on our multisite environment.

Open your `settings.php` files and comment out the following lines (comment these out in each `settings.php` file):

```
#$conf['memcache_servers'] = array('localhost:11211' => 'default');  
#$conf['cache_inc'] = './sites/all/modules/authcache/authcache.inc';
```

Now when you revisit your multisites, all of your tweaks and changes per site following the previous examples in this chapter should work without any issues.

Enabling page caching and CSS/JS optimization per site

With the Memcached disabled, you can visit your main site's performance admin page and enable page caching, compression, and CSS/JS optimization for each site. Then `site1` and `site2` could potentially run Drupal caching while you leave caching disabled on your localhost site.

In your `site1`, visit the main **Site configuration | Performance** page and set the page caching mode to **Normal**. Also, set a minimum cache lifetime, leave page compressions disabled, enable block cache, and optimize CSS and JavaScript files. Save configuration. Now you will have caching working on `site1`.

To run more advanced caching mechanisms in a multisite environment, you will need to review the configuration instructions for the respective module. For example, if you visit the Memcache API project page, you will see that there's an entire section devoted to Prefixing. If you have a multisite installation and you want to share the Memcached instance, you will need to add a unique prefix to each of the site's settings.php files in the \$conf section. We already commented this code out, so we would need to go back to our settings.php file and tweak the Memcache \$conf to make it look something like this:

```
$conf = array(  
    'memcache_key_prefix' => 'site1',  
) ;  
  
The conf array for site2 would look something like this:  
$conf = array(  
    'memcache_key_prefix' => 'site2',  
) ;
```

Earlier in the book, we discussed the Boost module. If you are using Boost to enhance performance on a multisite instance of Drupal, make sure to re-configure your Boost settings to support multisite. The Boost project page contains a wealth of resources and information on how to configure Boost for Drupal multisite. For example, if you are using Boost to cache directories and file extensions, you'll need to pay attention to the multisite information that the project page gives you regarding Boost directories and file extensions.

Multisite resources

If you continue using Drupal multisite, you will want to investigate how to configure multisite in various different environments, such as Linux, Mac, shared and dedicated servers. [Drupal.org](http://drupal.org) provides a wealth of resources on the Drupal multisite topic. There's an entire **Multi-site how-tos** page on drupal.org that contains links to various resources on Drupal. Everything, from installing multisite in 10 minutes to running multiple domains or virtual hosts using different databases, is covered. The Drupal multisite universe is a large one and will only become more flexible with the imminent release of Drupal 7.x. The Multisite resource guides are here on drupal.org: <http://drupal.org/node/43816>.

Summary

In this chapter, we installed and configured Drupal multisite in order to run multiple Drupal websites from the same core codebase, utilize contributed modules per multisite, and integrate with databases per site. We also discussed performance monitoring and caching in the Drupal multisite environment. You have successfully completed the following steps to configuring and setting up a Drupal multisite for fast performance:

- Installed Drupal multisite on Windows in a localhost development environment
- Set up multiple databases to support each site in our multisite
- Tweaked the `settings.php` file for each site in our multisite
- Edited the Apache configuration file to support `VirtualHosts`
- Tweaked the Windows hosts drivers to support our multiple sites and domain URLs
- Enabled modules and themes in each multisite site
- Discussed caching in a multisite environment
- Looked at multisite Drupal resources on `drupal.org`

You have successfully completed the *Drupal 6 Performance Tips* book and learned a wealth of information and resources about running your Drupal sites for faster optimized performance. To conclude, you should continue to work with Drupal daily and experiment with the myriad of performance, caching, and modular solutions that can help speed up your Drupal site and make it perform as well as possible. Drupal on!

Index

Symbols

.htaccess file

tweaking 78, 142-145

A

access denied (403) errors 84

advanced Boost configuration

about 157, 158

Boost Crawler settings 167, 168

database timestamp Boost rule, editing
160-162

database timestamp settings, checking 158

database timestamp settings, testing
162-166

event, adding 159

PHP filter module, enabling 159

rule, adding 159

Rules module, installing 158

Advanced cache module

about 193

block_cache.patch 194

comment_cache.patch 194

forum_cache.patch 194

installing 194

node_cache.patch 194

path_cache.patch 195

search_cache.patch 195

taxonomy_cache.patch 195

APC module 184, 197

Authcache

about 186, 212

Authcache Debug window, checking
192, 193

authcacheFooter code, checking 192

benefits 186, 187

configuring 188, 190

downloading 186

methods, for testing 192

page caching settings 190-192

settings.php file, tweaking 187

testing 192

B

block_cache.patch 194

blocks, Boost module

about 148

AJAX core statistics 149

pages cache configuration 149

pages cache status 149

Boost cache

clearing 148

Boost cacheability settings 139, 140

Boost configuration

about 137

testing 145, 146

Boost Crawler settings 167, 168

Boost directories 140, 142

Boost File Cache settings

about 138

Boost - HTML - Default minimum cache
lifetime 138

Boost - Static page cache 138

Gzip page compression setting 138

Boost installation 136

Boost module

about 135

basic configuration 151

blocks 148

Boost cache, clearing 148

- configuration, testing 145, 146
configuring 136
Global Redirect module 135
installing 136
Poormanscron, configuring 148
Poormanscron project 147
settings 137
- Boost settings**
about 137
Boost cacheability settings 139
Boost directories 140
Boost File Cache settings 138
file extensions 140, 142
HTACCESS file, tweaking 142
- C**
- Cache Router**
about 184
configuration 184
downloading 184
drawbacks 185
features 184
- cache technologies**
APC 184
database 184
eAccelerator 184
file 184
Memcache API 184
XCache 184
- caching, Drupal multisite** 212
- CCK module** 9
- CiviCRM** 53
- comment_cache.patch** 194
- contributed module**
updating 154
upgrading 38
- contributed module updates**
dealing with 22, 23
installing 19
- cron**
about 14
running 16, 21
- cron crawler** 167
- cron run** 158
- cron task**
Poormanscron module, installing 70
running 69
setting up, cPanel used 73
- CTools** 26
- D**
- database** 184
- DBlog**
about 82
access denied (403) errors 84
configuration 84
page not found (404) errors 84
- DB Maintenance module**
about 132
downloading 132
features 132-134
- Devel module**
about 9, 86
block, enabling 98, 99
block, using 100
configuring 87, 92-95
database queries, inspecting 95-98
downloading 87
download link 86
enabling 88
installing 87, 88
permissions, checking 88, 89
settings 92-95
Themer info tool, enabling 89-91
uses 86
- Devel module block**
cron, running 105
database queries 100
empty cache link 101
enabling 98, 99
function reference 103
Hook_elements() 103
menus, rebuilding 104
modules, reinstalling 105
PHP code, executing 101
PHPinfo() 104
session, viewing 105
theme developer, disabling/enabling 101
theme registry, cleaning 106
uses 100
Variable editor 106
variables, editing 106

- Devel module permissions**
 checking 88, 89
- Devel module settings**
 enabling 92
- Devel results**
 inspecting 95-98
- Drupal 5.19**
 upgrading steps 15, 16
- Drupal 5.19 upgrade**
 about 15
 contributed modules, dealing with 22, 23
 contributed module updates, installing 19
 cron, running 21
 recent log entries, checking 21
 Update Status module, installing 16-18
 Update Status module, uninstalling 20, 21
- Drupal 5.x core**
 contributed modules 9
 database, backing up 10-12
 site, backing up 10-12
 site, taking offline 12
 Status report, running 13, 14
 upgrading 8, 9
- Drupal 5.x core, contributed modules**
 CCK 9
 Devel 9
 FCK 9
 Imagecache 9
 Imagefield 9
 jQuery 9
 Lightbox 9
 Panels 9
 Views 1 9
 Webform 9
- Drupal 5.x upgrade plan** 9
- Drupal 5.x Views**
 backing up 23
 exporting 24, 26
 Panels code, reviewing 26, 27
- Drupal 6.13**
 downloading 29
 modules, disabling 58
 status report 52
 status report, checking 52
 themes, disabling 59
- Drupal 6.13 site**
 backing up, cPanel used 74-76
- Backing up, SFTP/FTP used 74-76
 database, backing up through
 phpMyAdmin 77
- Drupal 6.13 upgrade**
 contributed modules, disabling 28
 final prep checklist 27, 28
 Garland theme, enabling 29
- Drupal caching**
 about 59, 60
 block cache, enabling 62
 cache tables, in MySQL database 65
 configuring 64
 enabling 61
 Javascript file optimization, enabling 63
 normal caching, enabling 62
 page cache, enabling 61
 page compression, enabling 62
 performance cache, clearing 67
 theme registry, clearing 68
- Drupal configuration status**
 checking 52-54
- Drupal core**
 upgrading 30
- Drupal core upgrade**
 contributed modules, upgrading 38, 39
 PHP memory limit, upgrading 41-44
 steps 30, 32
 update.php, running 33-38
- Drupal Dblog module** 82
- Drupal multisite**
 about 200
 caching 212
 configuration, testing 209
 configuring, in localhost environment 201
 contributed modules, using 210
 core, using 210
 CSS/JS optimization, enabling 212, 213
 features 200
 modules, installing 211
 page caching, enabling 212, 213
 resources 213
 themes, installing 211
 themes, setting 211

E

eAccelerator 184

F

FCK module 9
file 184
Filecache module 197
file extensions, Boost 140, 142
forum_cache.patch 194

G

Garland theme 29
Global Redirect module
 about 135, 155
 checks, running 135
 configuring 155

H

Hook_elements() 103

I

i18n-ascii.txt file 157
Imagefield module 9

J

jQuery module 9

L

Lightbox2 module plugin 23
Lightbox module 9

M

Memcache
 running, without saving cache data 181
Memcache Admin module
 enabling 177
 Memcache statistics per page 179
 Memcache status page 178, 179
 Memcache tables, viewing in MySQL 181
Memcache API 184, 212
Memcache API and integration module
 about 170
 installing 175, 176
 Memcache, running, without saving cache data 181

Memcache Admin module, enabling 177
Memcached, integrating with PHP 5.2.x 173

Memcached libraries, installing 172
Memcached service, installing 172
Memcache tables, viewing in MySQL 181
MoWeS, installing 171
prerequisites, installing 170

Memcached

 about 212
 installing 170
 integrating, with PHP 5.2.x 173-175

Memcached libraries

 installing 172, 173

Memcached service

 installing 172, 173

MoWeS

 installing 171

MoWeS Portable development WAMP server

 171

multisite configuration, in localhost environment

 about 201
 Apache configuration, editing 206-208
 Base URL, tweaking 209
 databases, setting up 202-205
 hosts driver file, tweaking on Windows 208
 multisite folders, creating 202
 multisites, loading 209
 settings.php file, tweaking 205, 206
 testing 209, 210

MySQL Drupal database

 about 65
 cache_page table 66
 phpMyAdmin table 65, 66

N

node_cache.patch

P

page caching settings, Authcache

 about 190-192

page not found (404) errors

 84
 pages cache configuration block 150, 151
 pages cache status block 149

Panels 3.x module

Panels caching
about 127
panel, adding to content 127-129
panel, creating 127-129
using 127

Panels module 9

path_cache.patch 195

Pathauto module 156, 157

Photo Gallery view 23

PHP and MySQL configuration settings
checking 55, 56

PHPinfo() 104

PHP memory limit
upgrading 41

phpMyAdmin 10

Plesk 10

Poormanscron
about 147
configuring 148
project page 147

Poormanscron module
about 70
installing 70-72

R

recent log entries
checking 22
viewing 82, 83

S

search_cache.patch 195

status report, Drupal 6.13
checking 52
files, removing 57

PHP and MySQL settings, checking 55

Update Status module, enabling 57, 58

T

taxonomy_cache.patch
about 195
downloading 195
enabling 195, 196
installing 195

test categories
generating 118, 119

test content
generating 119, 120

test users
generating 115-117

Themer info tool, Devel module
enabling 89-91

Throttle module
about 110
blocks, throttling 114, 115
configuration, accessing 110
configuring 110
configuring, for auto throttling features 111, 112
enabling 110
modules, throttling 113, 114

Transliteration module 156, 157

U

update.php
running 33-38

updated Zen theme files
installing 44

updated Zen theme files installation
about 44
custom theme, upgrading 44-46
site, placing online 49, 50
Views, cleaning up 47, 48
Views, resetting 47, 48

Update Status module
enabling 57, 58
installing 16-18
uninstalling 20, 21

V

Variable editor 106

Views
caching 120-124

Views 1 module 9

Views 2 module 120

Views 2 module cache
clearing 124, 125

Views module 24

W

WAMP 171
Webform module 9

Z

Zen StarterKit theme 9

X

XCache 184



Thank you for buying Drupal 6 Performance Tips

Packt Open Source Project Royalties

When we sell a book written on an Open Source project, we pay a royalty directly to that project. Therefore by purchasing Drupal 6 Performance Tips, Packt will have given some of the money received to the Drupal project.

In the long term, we see ourselves and you – customers and readers of our books – as part of the Open Source ecosystem, providing sustainable revenue for the projects we publish on. Our aim at Packt is to establish publishing royalties as an essential part of the service and support a business model that sustains Open Source.

If you're working with an Open Source project that you would like us to publish on, and subsequently pay royalties to, please get in touch with us.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

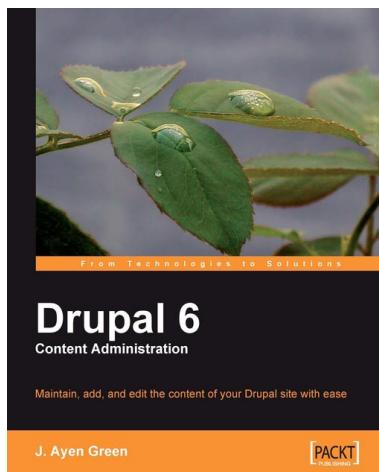
We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

About Packt Publishing

Packt, pronounced 'packed', published its first book "Mastering phpMyAdmin for Effective MySQL Management" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.PacktPub.com.



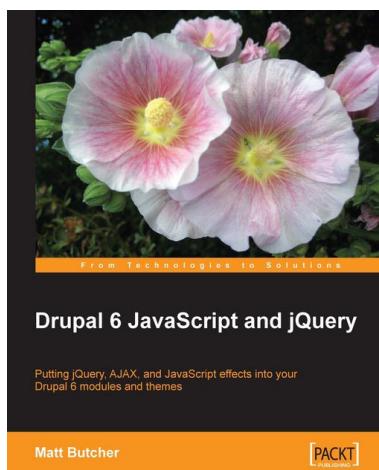
Drupal 6 Content Administration

ISBN: 978-1-847198-56-3

Paperback: 196 pages

Maintain, add to, and edit content of your Drupal site with ease

1. Keep your Drupal site up to date: easily edit, add to, and maintain your site's content, even if you've never used Drupal before!
2. Covers the full range of content that you might want on your site: richly formatted text, images, videos, as well as blog posts, calendar events, and more
3. Get to grips with managing users, slaying spam, and other activities that will help you maintain a content-rich site
4. Concise, targeted information with easy-to-follow hands-on examples



Drupal 6 JavaScript and jQuery

ISBN: 978-1-847196-16-3

Paperback: 340 pages

Putting jQuery, AJAX, and JavaScript effects into your Drupal 6 modules and themes

1. Learn about JavaScript support in Drupal 6
2. Packed with example code ready for you to use
3. Harness the popular jQuery library to enhance your Drupal sites
4. Make the most of Drupal's built-in JavaScript libraries

Please check www.PacktPub.com for information on our titles

[Download at Wow! eBook](#)



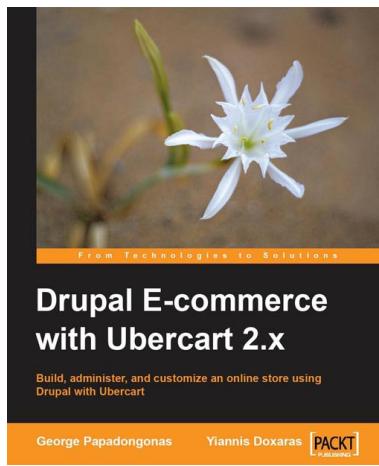
Drupal 6 Site Blueprints

ISBN: 978-1-847199-04-1

Paperback: 272 pages

Ready-made plans for 12 different professional Drupal sites

1. Instant Drupal - Build 12 exciting and simple web projects
2. Expand and tailor the sample projects to your client's need
3. Create quick prototypes of commonly used applications within hours
4. Develop your own custom application by merging features from the example projects
5. Apply easy methods to optimize the performance of your site



Drupal E-commerce with Ubercart 2.x

ISBN: 978-1-847199-20-1

Paperback: 300 pages

Build, administer, and customize an online store using Drupal with Ubercart

1. Create a powerful e-shop using the award-winning CMS Drupal and the robust e-commerce module Ubercart
2. Create and manage the product catalog and insert products in manual or batch mode
3. Apply SEO (search engine optimization) to your e-shop and adopt turn-key internet marketing techniques
4. Implement advanced techniques like cross-selling, product comparison, coupon codes, and segmented pricing

Please check www.PacktPub.com for information on our titles