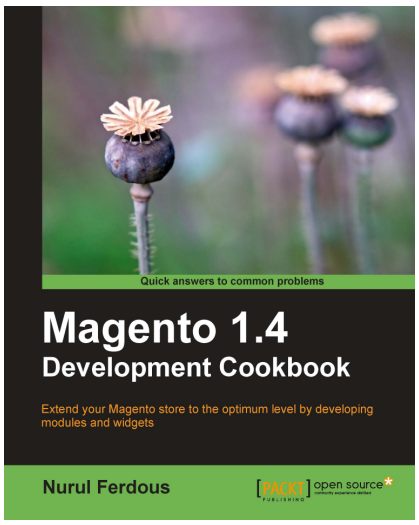
[PACKT] open source
community experience distilled
PUBLISHING

Magento 1.4 Development Cookbook

Nurul Ferdous



Chapter No. 11
"Performance Optimization"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.11 "Performance Optimization"

A synopsis of the book's content

Information on where to buy this book

About the Author

Nurul Ferdous is an open source enthusiast and IT specialist from Bangladesh who is currently working for TM Secure Inc. as a LAMP consultant. In fact, he is a soldier turned programmer. He started his career with the Bangladesh Air Force. He has also served in RAB as an intelligence staff where he was nominated for the President's Police medal for his contribution to national security. He is a true passionate programmer. He started his run on software development back in 2004, while he was working in the Bangladesh Air Force.

His primary skills are as a PHP developer. He is a Zend Certified PHP 5 Engineer, and contributes to a number of PHP projects, blogs on PHP-related topics, and presents talks and tutorials related to PHP development and the projects to which he contributes. He also contributes on open source community regularly. He is also a certified professional on TDD and Code Refactoring.

He has served in some top notch software companies both at home and abroad such as BIPL, Right Brain Solutions Ltd., TM Secure Inc., NameDepot.com Inc., and so on as a programmer, software engineer, and consultant. He also writes at his personal blog `http://dynamicguy.com` when he is not baking with codes.

The very first person whom I would like to thank who made this happen is Dilip Venkatesh along with Meeta Rajani, Aditi Suvarna, and all PACKT personnels who worked on this book. I am also thankful to my wife, Ferdousy Chowdhury and my kid, Riva. They have helped me a lot during the whole writing process!

Magento 1.4 Development Cookbook

Magento is the fastest growing PHP-based eCommerce solution based on the Zend Framework. This robust CMS helps developers to enrich their store with more useful functionalities with custom modules. Developing a Magento store to get the desired look and feel is not as easy as you might believe, and may take hours due to the wealth of features available for you.

This book will provide unparalleled guidelines to develop a much faster and captivating Magento store by writing custom modules and powerful customizations. This book covers everything from common development tasks to customization of your store as per your needs.

If you choose to work through all the recipes from the beginning, you will have a development platform ready to work with Magento. You will also explore different ways to customize the look and feel of a Magento store to facilitate your customers and create a better user experience. Integration of Magento with WordPress to add striking functionality to your store will be accomplished in just a few steps. The Magento security measures have been taken care of in some cool recipes by editing the parameters in the admin panel. Setting up a master slave setup for Magento database is discussed along with other database optimizations in the database section. Developing new modules and widgets for Magento is elaborately described in many recipes. Magento's performance optimization is the most important part of this book, which is armed with some easy and incredible recipes dealing with YSlow, Page Speed, Siege, Apache bench, Apache configuration, php.ini optimization, and caching with APC or Memcached. The work procedure behind the wall is explained in an easy manner so that both a novice as well as an experienced developer could benefit from it. This book also has some recipes which are not only useful for Magento but also for any other LAMP-based project.

What This Book Covers

Chapter 1, Getting Started with Magento Development, introduces Magento and describes how to set up different tools required for a Magento project. It also illustrates how to work with Magento code.

Chapter 2, CMS and Design, describes how to customize the look and feel of a Magento store to change the default appearance of Magento to satisfy our corporate identity.

Chapter 3, Adding Extra Functionalities, illustrates how to design an eCommerce store in a user-friendly way so that any customer feels comfortable while browsing the shop, by adding some cool functionalities in the shop.

Chapter 4, Customizing a Store, focuses on some of the vulnerable features that are a must for any successful online store such as customizing Twitter handle field, deleting orders in Magento, creating Catalog and Shopping Cart Price Rules, among others.

Chapter 5, Playing with Products, helps you set up some important settings in Magento to enhance the usability of your store such as adding a Facebook 'Like' button in the product page, embedding a video in the product details page, and so on.

Chapter 6, Adding a Professional Touch to Your Site, covers topics such as installing Magento 1.4 in PHP 5.3.2 (without mcrypt), optimizing Magento store for search engines, among others.

Chapter 7, Database Design, covers the resources and database connections, database replication using Master Slave setup, and also explains how to work with Magento's EAV design.

Chapter 8, Creating a Module, covers how to create a module, design a template, and set up SQL for the module.

Chapter 9, Creating a Shipping Module, describes the steps on how to create the configuration file for a new shipping module along with adding a module in the backend and the frontend.

Chapter 10, Writing a Social Widget, explains how to make your own widget and also share this widget with various social network links.

Chapter 11, Performance Optimization, describes how to ensure the optimum performance of your Magento store and overcome the most common pitfalls by applying the techniques described in the different recipes in this chapter.

Chapter 12, Debugging and Unit Testing, guides you through installing, configuring, and using Xdebug, Zend Wildfire, and PHPUnit in an easy way.

11
Performance Optimization

In this chapter, we will cover:

- Measuring/benchmarking your Magento with Siege, ab, Magento profiler, YSlow, Page Speed, GTmetrix, and WebPagetest
- Optimizing Magento database and MySQL configuration
- Optimizing web server configuration (Apache)
- Tuning Magento configurations
- Using APC/Memcached as the cache backend
- Accelerating PHP: php.ini configuration
- Applying YSlow and Page Speed rules

Introduction

Users really respond to speed.—Marissa Mayer, Google vice president of the research section and user experience.

We will explain why this quote is true throughout this chapter. Her key insight for the crowd at the Web 2.0 Summit is that "slow and steady doesn't win the race". Today people want fast and furious. Not convinced? Okay, let's have a look at some arguments:

- 500ms lost is to lose 20 percent of traffic for Google (this might be why there are only ten results per page in the research).
- Increased latency of 100ms costs 1 percent of sales for Amazon.

- Reducing by 25 percent the weight of a page is to win 25 percent of users in the medium term for Google.
- Losing 400ms is to have a 5-9 percent drop in addition to Yahoo!, an editorial site.

As we can see, this is the era of milliseconds and terabytes, so we have to pay a big price if we can't keep up. This chapter will ensure the optimum performance of your Magento store and overcome some of the most common pitfalls that people encounter. For example, in Magento, backoffice is slow, frontend tends to die while loading, accessibility is far from perfect, an AJAX request is not successful due to expiration of the session, nothing happens, no error message, just nothing, the nightmare of XML layout, and so on.

This chapter will be very helpful whether you are a novice or a pro on performance optimization. Follow every single chapter and your Magento store should get a notable performance raise.

By applying the techniques described in different recipes in this chapter, you can achieve a tremendous performance boost (approximately 300 percent) on any LAMP application.

Measuring/benchmarking your Magento with Siege, ab, Magento profiler, YSlow, Page Speed, GTmetrix, and WebPagetest

The very first task of any website's performance optimization is to know its pitfalls. In other words, know why it is taking too much time. Who are the culprits? Fortunately, we have some amicable friends who will guide us through. Let's list them:

- **ab** (**ApacheBench**): This is bundled with every Apache as a benchmarking utility.
- **Siege**: This is an open source stress/regression test and benchmark utility by *Joe Dog*.
- **Magento profiler**: This is a built-in Magento profiler.
- **YSlow**: This is a tool from Yahoo! We have been using it for years. This is in fact a firebug add-on.
- **Page Speed**: This is yet another firebug add-on from Google to analyze page performance on some common rules.
- **GTmetrix**: This is a cool online web application from which you can get both YSlow and Page Speed in the same place. Opera fanboys who don't like Firefox or Chrome might use it for YSlow and Page Speed here.
- **WebPagetest**: This is another online tool for benchmarking as GTmetrix. It also collects and shows screenshots with the reports.

Okay, we are introduced to our new friends. In this recipe, we will work with them and find the pitfalls of our Magento store, which will be addressed throughout the whole chapter. Let's play!

Getting ready

Before starting the work, we have to make sure that every required tool is in place. Let's check it.

ab: This Apache benchmarking tool is bundled with every Apache installation. If you are on a Linux-based distribution, you can give it a go by issuing the following command in the terminal:

```
ab -h
```

Siege: We will use this tool in the same box as our server. So make sure you have it installed. You can see it by typing this command (note that the option is capital V):

```
siege -V
```

If it's installed, you should see the installed version information of Siege. If it's not, you can install it with the following command in any Debian-based Distro:

```
sudo apt-get install siege
```

You can also install it from source. To do so, grab the latest source from here: `ftp://ftp.joedog.org/pub/siege/siege-latest.tar.gz`, then issue the following steps sequentially:

```
# go the location where you downloaded siege
tar xvzf siege-latest.tar.gz
# go to the siege folder. You should read it with something like
siege-2.70
./configure
make
make install
```

If you are on a Windows-based box, you would find it as:

```
apache/bin/ab.exe
```

Magento Profile: This is also a built-in tool with Magento.

YSlow: This firebug add-on from Firefox could be installed via the Internet from here: `http://developer.yahoo.com/yslow/`. Firebug add-on is a dependency for YSlow.

Page Speed: This is also a firebug add-on that can be downloaded and installed from: `http://code.google.com/speed/page-speed/download.html`.

For using GTmetrix and WebPagetest, we will need an active Internet connection. Make sure you have these.

How to do it...

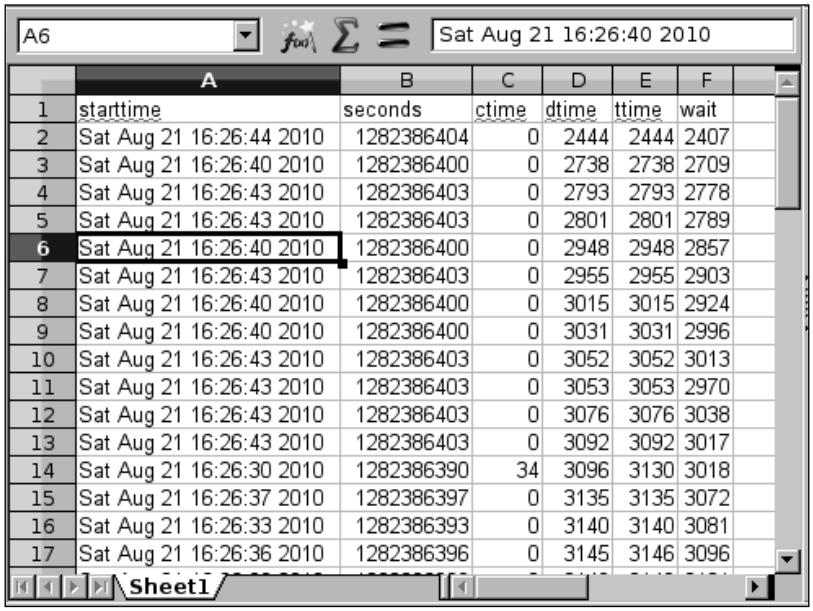
Using the simple tool ab:

1. If you are on a Windows environment, go to the `apache/bin/` folder and if you are on Unix, fire up your terminal and issue the following command:

   ```
   ab -c 10 -n 50 -g mage.tsv http://magento.local.com/
   ```

2. In the previous command, the params denote the following:
   - **-c**: This is the concurrency number of multiple requests to perform at a time. The default is one request at a time.
   - **-n**: This requests the number of requests to perform for the benchmarking session. The default is to just perform a single request, which usually leads to non-representative benchmarking results.
   - **-g** (**gnuplot-file**): This writes all measured values out as a gnuplot or **TSV** (**tab separate values**) file. This file can easily be imported into packages like Gnuplot, IDL, Mathematica, Igor, or even Excel. The labels are on the first line of the file.

3. The preceding command generates some benchmarking report in the terminal and a file named `mage.tsv` in the current location, as we specified in the command.
4. If we open the `mage.tsv` file in a spreadsheet editor such as Open Office or MS Excel, it should read as follows:



A6 | Sat Aug 21 16:26:40 2010

	A	B	C	D	E	F
1	starttime	seconds	ctime	dtime	ttime	wait
2	Sat Aug 21 16:26:44 2010	1282386404	0	2444	2444	2407
3	Sat Aug 21 16:26:40 2010	1282386400	0	2738	2738	2709
4	Sat Aug 21 16:26:43 2010	1282386403	0	2793	2793	2778
5	Sat Aug 21 16:26:43 2010	1282386403	0	2801	2801	2789
6	Sat Aug 21 16:26:40 2010	1282386400	0	2948	2948	2857
7	Sat Aug 21 16:26:43 2010	1282386403	0	2955	2955	2903
8	Sat Aug 21 16:26:40 2010	1282386400	0	3015	3015	2924
9	Sat Aug 21 16:26:40 2010	1282386400	0	3031	3031	2996
10	Sat Aug 21 16:26:43 2010	1282386403	0	3052	3052	3013
11	Sat Aug 21 16:26:43 2010	1282386403	0	3053	3053	2970
12	Sat Aug 21 16:26:43 2010	1282386403	0	3076	3076	3038
13	Sat Aug 21 16:26:43 2010	1282386403	0	3092	3092	3017
14	Sat Aug 21 16:26:30 2010	1282386390	34	3096	3130	3018
15	Sat Aug 21 16:26:37 2010	1282386397	0	3135	3135	3072
16	Sat Aug 21 16:26:33 2010	1282386393	0	3140	3140	3081
17	Sat Aug 21 16:26:36 2010	1282386396	0	3145	3146	3096

Sheet1

5. You can tweak the ab params and view a full listing of params by typing `ab -h` in the terminal.

Using Siege:

1. Let's lay Siege to it! Siege is an HTTP regression testing and benchmarking utility. It was designed to let web developers measure the performance of their code under duress, to see how it will stand up to load on the Internet. Siege supports basic authentication, cookies, HTTP, and HTTPS protocols. It allows the user to hit a web server with a configurable number of concurrent simulated users. These users place the web server 'under Siege'.
2. Let's create a text file with the URLs that would be tested under Siege. We can pass a single URL in the command line as well. We will use an external text file to use more URLs through a single command. Create a new text file in the terminal's current location. Let's assume that we are in the `/Desktop/mage_benchmark/` directory. Create a file named `mage_urls.txt` here and put the following URLs in it:

```
http://magento.local.com/
http://magento.local.com/skin/frontend/default/default/favicon.ico
http://magento.local.com/js/index.php?c=auto&f=,prototype/
prototype.js,prototype/validation.js,scriptaculous/builder.
js,scriptaculous/effects.js,scriptaculous/dragdrop.
js,scriptaculous/controls.js,scriptaculous/slider.js,varien/
js.js,varien/form.js,varien/menu.js,mage/translate.js,mage/
cookies.js
http://magento.local.com/skin/frontend/default/default/css/print.
css
http://magento.local.com/skin/frontend/default/default/css/styles-
ie.css
http://magento.local.com/skin/frontend/default/default/css/styles.
css
http://magento.local.com/skin/frontend/default/default/images/np_
cart_thumb.gif
http://magento.local.com/skin/frontend/default/default/images/np_
product_main.gif
http://magento.local.com/skin/frontend/default/default/images/
np_thumb.gif
http://magento.local.com/skin/frontend/default/default/images/
slider_btn_zoom_in.gif
http://magento.local.com/skin/frontend/default/default/images/
slider_btn_zoom_out.gif
http://magento.local.com/skin/frontend/default/default/images/
spacer.gif
http://magento.local.com/skin/frontend/default/default/images/
media/404_callout1.jpg
http://magento.local.com/electronics/cameras.html
http://magento.local.com/skin/frontend/default/default/images/
```

```
media/furniture_callout_spot.jpg
http://magento.local.com/skin/adminhtml/default/default/boxes.css
http://magento.local.com/skin/adminhtml/default/default/ie7.css
http://magento.local.com/skin/adminhtml/default/default/reset.css
http://magento.local.com/skin/adminhtml/default/default/menu.css
http://magento.local.com/skin/adminhtml/default/default/print.css
http://magento.local.com/nine-west-women-s-lucero-pump.html
```

3. These URLs will vary with yours. Modify it as it fits. You can add more URLs if you want.
4. Make sure that you are in the `/Desktop/mage_benchmark/` directory in your terminal. Now issue the following command:

   ```
   siege -c 50 -i -t 1M -d 3 -f mage_urls.txt
   ```

5. This will take a fair amount of time. Be patient. After completion it should return a result something like the following:

   ```
   Lifting the server siege..      done.
   Transactions:                    603 hits
   Availability:                  96.33 %
   Elapsed time:                  59.06 secs
   Data transferred:              10.59 MB
   Response time:                  1.24 secs
   Transaction rate:              10.21 trans/sec
   Throughput:                     0.18 MB/sec
   Concurrency:                   12.69
   Successful transactions:         603
   Failed transactions:              23
   Longest transaction:           29.46
   Shortest transaction:           0.00
   ```

6. Repeat the steps 1 and 3 to produce reports with some variations and save them wherever you want.
7. The option details could be found by typing the following command in the terminal:

   ```
   siege -h
   ```

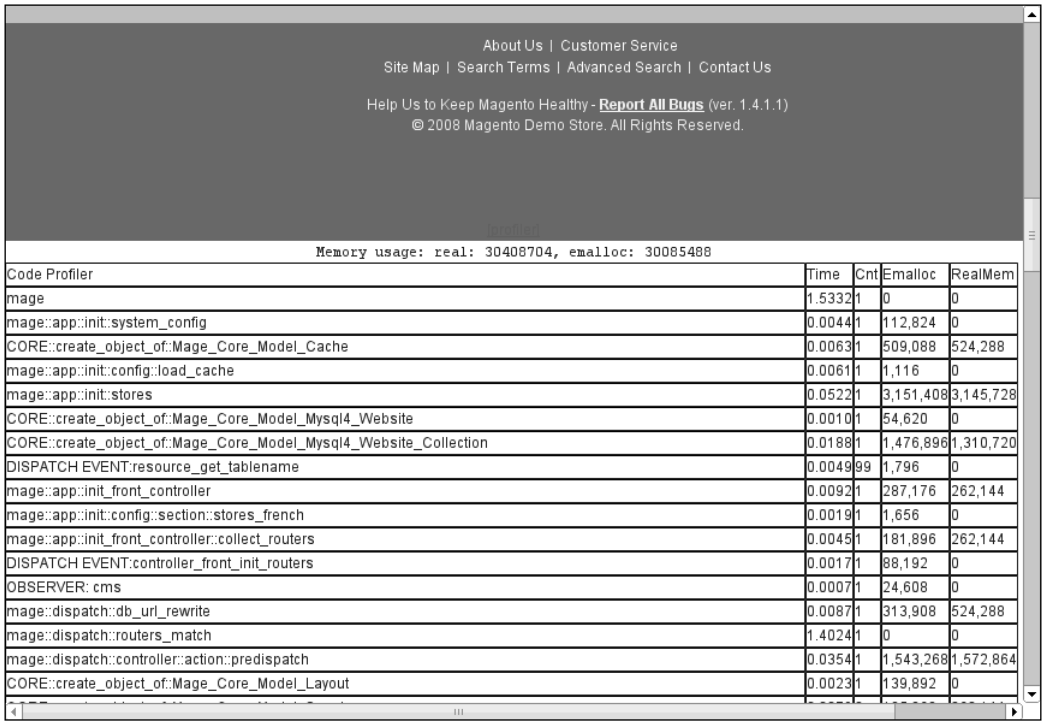
Magento profiler:

1. Magento has a built-in profiler. You can enable it from the backend's **System** | **Configuration** | **Advanced** | **Developer** | **Debug** section.

2. Now open the `index.php` file from your Magento root directory. Uncomment line numbers 65 and 71. The lines read as follows:

```
line 65: #Varien_Profiler::enable(); // delete #
line 71: #ini_set(‹display_errors›, 1); // delete #
```

3. Save this file and reload your Magento frontend in the browser. You should see the profiler data at the bottom of the page, similar to the following screenshot:



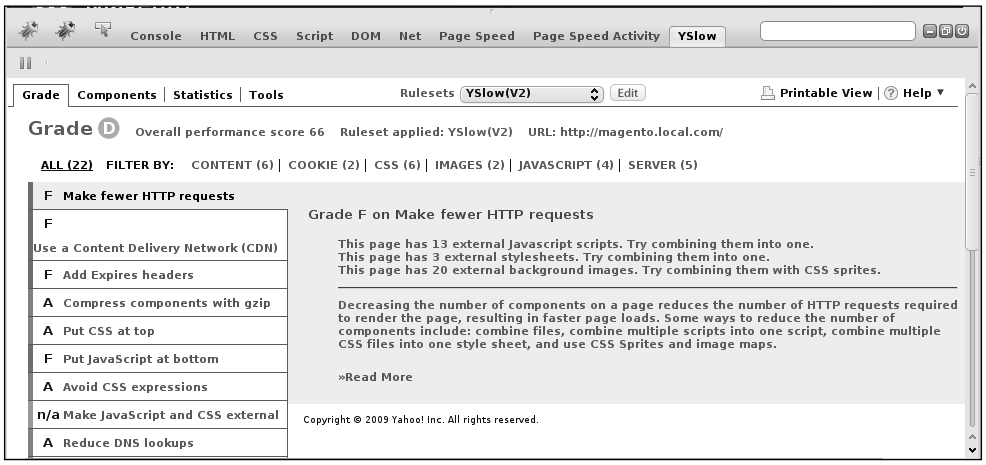
Memory usage: real: 30408704, emalloc: 30085488

Code Profiler	Time	Cnt	Emalloc	RealMem
mage	1.5332	1	0	0
mage::app::init::system_config	0.0044	1	112,824	0
CORE::create_object_of::Mage_Core_Model_Cache	0.0063	1	509,088	524,288
mage::app::init::config::load_cache	0.0061	1	1,116	0
mage::app::init::stores	0.0522	1	3,151,408	3,145,728
CORE::create_object_of::Mage_Core_Model_Mysql4_Website	0.0010	1	54,620	0
CORE::create_object_of::Mage_Core_Model_Mysql4_Website_Collection	0.0188	1	1,476,896	1,310,720
DISPATCH EVENT:resource_get_tablename	0.0049	99	1,796	0
mage::app::init_front_controller	0.0092	1	287,176	262,144
mage::app::init::config::section::stores_french	0.0019	1	1,656	0
mage::app::init_front_controller::collect_routers	0.0045	1	181,896	262,144
DISPATCH EVENT:controller_front_init_routers	0.0017	1	88,192	0
OBSERVER: cms	0.0007	1	24,608	0
mage::dispatch::db_url_rewrite	0.0087	1	313,908	524,288
mage::dispatch::routers_match	1.4024	1	0	0
mage::dispatch::controller::action::predispatch	0.0354	1	1,543,268	1,572,864
CORE::create_object_of::Mage_Core_Model_Layout	0.0023	1	139,892	0

Yslow:

1. We have already installed the YSlow firebug add-on. Open the Firefox browser and let's activate it by pressing the *F12* button or clicking the firebug icon from the bottom-right corner of Firefox.
2. Click on the **YSlow** link in firebug.
3. Select the **Rulesets**. In my case I chose **YSlow (V2)**.
4. Click on the **Run Test** button.

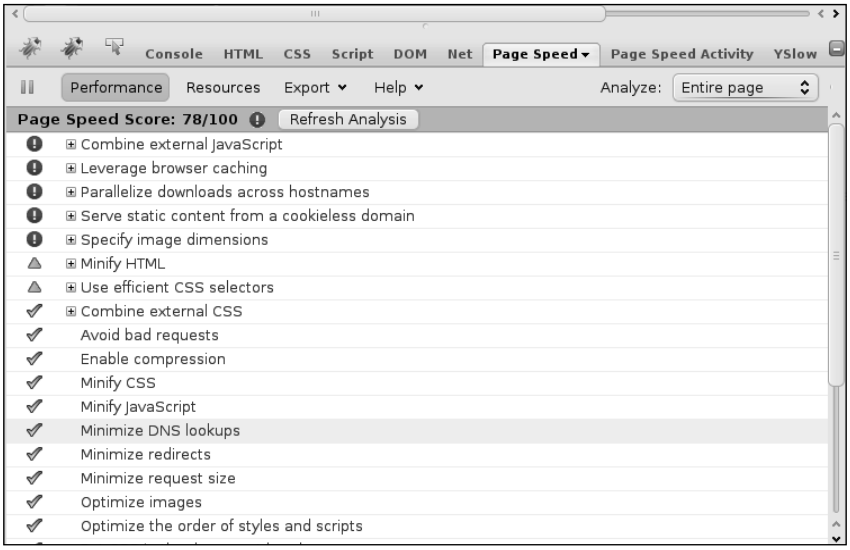
5. After a few seconds you will see a report page with the grade details. Here is mine:



6. You can click on the links and see what it says.

Page Speed:

1. Fire up your Firefox browser.
2. Activate the firebug panel by pressing *F12*.
3. Click on the **Page Speed** link.
4. Click on the **Performance** button and see the **Page Speed Score** and details. The output should be something like the following screenshot:



Using GTmetrix:

This is an online tool to benchmark a page with a combination of YSlow and Page Speed. Visit `http://gtmetrix.com/` and DIY (Do It Yourself).

Using WebPagetest:

This is a similar tool as GTmetrix, which can be accessed from here: `http://www.webpagetest.org/`.

How it works...

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

The analysis that Siege leaves you with can tell you a lot about the sustainability of your code and server under duress. Obviously, availability is the most critical factor. Anything less than 100 percent means there's a user who may not be able to access your site. So, in the above case, there's some issue to be looked at, given that availability was only 96.33 percent on a sustained 50 concurrent, one minute user Siege.

Concurrency is measured as the time of each transaction (defined as the number of server hits including any possible authentication challenges) divided by the elapsed time. It tells us the average number of simultaneous connections. High concurrency can be a leading indicator that the server is struggling. The longer it takes the server to complete a transaction while it's still opening sockets to deal with new traffic, the higher the concurrent traffic and the worse the server performance will be.

Yahoo!'s exceptional performance team has identified 34 rules that affect web page performance. YSlow's web page analysis is based on the 22 of these 34 rules that are testable. We used one of their predefined ruleset. You can modify and create your own as well.

When analyzing a web page, YSlow deducts points for each infraction of a rule and then applies a grade to each rule. An overall grade and score for the web page is computed by summing up the values of the score for each rule weighted by the rule's importance. Note that the rules are weighted for an average page. For various reasons, there may be some rules that are less important for your particular page.

In YSlow 2.0, you can create your own custom rulesets in addition to the following three predefined rulesets:

- YSlow(V2): This ruleset contains the 22 rules
- Classic(V1): This ruleset contains the first 13 rules
- Small Site or Blog: This ruleset contains 14 rules that are applicable to small websites or blogs

Page Speed generates its results based on the state of the page at the time you run the tool. To ensure the most accurate results, you should wait until the page finishes loading before running Page Speed. Otherwise, Page Speed may not be able to fully analyze resources that haven't finished downloading.

Windows users can use Fiddler as an alternative to Siege. You can download it from `http://www.fiddler2.com/fiddler2/`, which is developed by Microsoft.

Optimizing Magento database and MySQL configuration

We already know the bottlenecks of our Magento store. Let us start with the Magento database and our MySQL server as it impacts drastically on the overall performance. In this recipe, we will optimize our store database as well as our MySQL server.

How to do it...

Optimizing the Magento database:

1. This is a simple task. Launch your MySQL administration tool. phpMyAdmin is fine. Select your Magento database.
2. You should see there are two pages for table structures. Click on the **check all** button from the bottom.
3. Now keeping all tables selected, click on the **Repair** table from the drop-down besides the **check all** link.
4. Do the same for the **Optimize** table from the drop-down.
5. Repeat these steps for page 2 as well.
6. At this stage, our database is good to go.
7. Replicate your MySQL server as instructed in *Chapter 7*, *Database Design*, the *Magento database replication using Master Slave setup* recipe
8. Split the write and read to master and slave database. Refer to *Chapter 7*, *Database Design*, the *Magento database replication using Master Slave setup* recipe.

Optimizing MySQL server:

1. Let us start from the system's operating system itself. To get the best use of multiple-CPU machines, you should use Linux (because 2.4 and later kernels have good SMP support). File systems such as ReiserFS and XFS do not have any size 2

GB limitation that is on ext2 though; ext2 could be patched for more than 2 GB size with the LFS patch. For running Magento, it's highly recommended to use a dedicated server or at least a VPS.

2. If you have enough RAM, you could remove all swapped devices. Some operating systems use a swap device in some contexts even if you have free memory.
3. Avoid external locking by using skip-external-locking in the `[mysqld]` section of `my.cnf`.
4. The next important task is to set the `key_buffer_size` for MyISAM engine. Log in to your MySQL server with root access in the terminal.

   ```
   mysql> SHOW VARIABLES LIKE '%key_buffer%';
   ```

5. You should specify the current allocation for `key_buffer_size`. It is used by MyISAM tables to cache Index only, not data. If you have got a MyISAM system only, then the `key_buffer_size` should be 30 percent of memory. Remember that there is a limit of 4 GB per key buffer. MyISAM tables are used for temporary tables anyway. Let's set it as 512M. So, the following will be the MySQL command:

   ```
   mysql> SET GLOBAL key_buffer_size = 536870912;
   ```

6. We will set the rest of the configurations via the `my.cnf` file from the `/etc/mysql/` directory.
7. Open it and create the `[mysqld]` section as follows:

   ```
   [mysqld]
   key_buffer = 512M
   max_allowed_packet = 64M
   thread_stack = 192K
   thread_cache_size = 32
   table_cache = 512
   query_cache_type = 1
   query_cache_size = 52428800
   tmp_table_size = 128M
   expire_logs_days = 10
   max_binlog_size = 100M
   sort_buffer_size = 4M
   read_buffer_size = 4M
   read_rnd_buffer_size = 2M
   myisam_sort_buffer_size = 64M
   wait_timeout = 300
   max_connections = 400
   ```

8. Save and restart your MySQL server.

How it works...

The most important tuning for any MySQL server are `key_buffer`, `query_cache` and `table_cache`. Make sure that you have specified the appropriate unit and value. Skip-external-locking is also an important tuning. It helps to deny any external locking.

You can view the current MySQL server status and variables by issuing the following commands:

Command	Description
`mysql> SHOW STATUS;`	This shows the current MySQL server status. It is available since MySQL 5.0, and `SHOW STATUS` now defaults to this. Some variables are global only; they will still be shown in the `SHOW STATUS` output.
`mysql> SHOW VARIABLES;`	This shows the MySQL variables.
`mysql> SHOW INNODB STATUS;`	This shows the current `InnoDB` status.
`mysql> SHOW GLOBAL STATUS;`	This shows the global server status load. It is good for understanding the load.
`mysql> SHOW LOCAL STATUS;`	This is great for query/transaction profiling.
`mysql> mysqladmin extended -i100 -r`	This is great to sample what is happening with MySQL Server Now.

You are encouraged to benchmark your server while applying these tunings. You should get about a 50 percent performance gain after these settings are applied.

There's more

Some wrong units in MySQL server tuning:

- `table_cache=128M`

  Wrong, table cache is measured in entries

- `key_buffer_size=1024`

  Wrong again, key buffer should be specified in bytes

- `innodb_max_dirty_pages_pct=8G`

  This one is set in percents

Optimizing Apache web server configuration

Now-a-days there are some other alternatives for Apache, but still the majority of us use Apache as a web server. The ultimate performance of any web application depends on some other factors as well such as CPU, network card, disk, OS, and RAM.

The most important hardware that affects any web server's performance is RAM. The more the better. In this recipe, we will optimize the web server, preferably a DS or at least a VPS.

How to do it...

1. Update the operating system to its latest stable release. In most cases, OS suppliers have introduced significant performance improvements to their TCP stacks and thread libraries.
2. Update your sendfile(2) system call support with the latest patch if your box is Linux and its kernel is lower than 2.4. If your box has got a kernel higher than 2.4 then it's ok. This ensures Apache 2 to deliver static contents faster with lower CPU utilization.
3. Keep other background applications minimum. For example, in Unix, switch off NFS, any printing services, and even sendmail if it's not needed. Under Windows, use the system control panel to optimize the system for applications and system cache, and optimize the system for performance.
4. ReiserFS and XFS are some good filesystems to use for better disk I/O.
5. Web server should never ever have to swap, as swapping increases the latency of each request. You should control the MaxClients setting so that your server does not spawn so many children, that it starts swapping. Determine the size of your average Apache process, by looking at your process list via a tool such as top, and divide this into your total available memory, leaving some room for other processes.
6. Set **HostnameLookups** to **Off**, if your Apache is lower than 1.3. If Apache is higher than 1.3, then **HostnameLookups** is defaulted to **Off**.
7. It is highly discouraged to use `SymLinksIfOwnerMatch`, you can rather use it as `Options FollowSymLinks + SymLinksIfOwnerMatch` for specific directories. And for other locations use `Options FollowSymLinks`. This will prevent `lstat(2)` system calls. By the way, the results of `lstat()` are never cached.
8. Never use `DirectoryIndex` as a wildcard, such as the following:

   ```
   DirectoryIndex index
   ```

9. Rather, use a complete one such as the following:

   ```
   DirectoryIndex index.php index.html index.cgi index.pl index.shtml
   ```

10. Enable mod deflate and mod header with the following command:

    ```
    sudo a2enmod deflate
    sudo a2enmod header
    ```

11. Let us add some deflate settings in our `.htaccess` of our Magento store. Find and open the `.htaccess` from Magento root and modify it to show the deflate block as follows:

    ```
    <IfModule mod_deflate.c>

    ##############################################
    ## enable apache served files compression
    ## http://developer.yahoo.com/performance/rules.html#gzip
    ```

```
    # Insert filter on all content
    SetOutputFilter DEFLATE
    # Insert filter on selected content types only
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
text/css text/javascript

    # Netscape 4.x has some problems...
    BrowserMatch ^Mozilla/4 gzip-only-text/html

    # Netscape 4.06-4.08 have some more problems
    BrowserMatch ^Mozilla/4\.0[678] no-gzip

    # MSIE masquerades as Netscape, but it is fine
    BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

    # Don›t compress images
    SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-
vary

    # Make sure proxies don›t deliver the wrong content
    Header append Vary User-Agent env=!dont-vary

</IfModule>
```

12. The next task is to set up mod expires. Issue the following command in your Linux terminal:

    sudo a2enmod expires

    sudo /etc/init.d/apache2 restart

13. We must add expires settings in our `.htaccess`. Find the `expires` block and make it as follows:

```
<IfModule mod_expires.c>

###############################################
## Add default Expires header
## http://developer.yahoo.com/performance/rules.html#expires

    ExpiresDefault «access plus 1 year»
    ExpiresActive On
    ExpiresByType image/gif «access plus 30 days»
    ExpiresByType image/jpg «access plus 30 days»
    ExpiresByType image/jpeg «access plus 30 days»
    ExpiresByType image/png «access plus 30 days»
```

```
    ExpiresByType image/x-icon «access plus 30 days»
    ExpiresByType text/css «access plus 30 days»
    ExpiresByType application/x-javascript «access plus 30 days»

</IfModule> Apache 1.1 comes with the Keep-Alive support on by
default.
```

14. Apache 1.1 and higher has come up with it's `KeepAlive` directive as `On` by default. Make sure that the `KeepAlive` is `On`. This is a trick where multiple HTTP requests can be funneled through a single TCP connection. You can modify it from the `/etc/apache2/apache2.conf` file.
15. Set up your MPM as best. Experiment with it. Here is my configuration, this might and should be different depending on your system's resources:

```
StartServers          50
MinSpareServers       15
MaxSpareServers       30
MaxClients            225
MaxRequestsPerChild  4000
```

16. Benchmark your Magento again and see your changes in action!

How it works...

The overall performance of a LAMP application depends on the factors described previously and obviously your application itself. If your application cannot serve 100 concurrent hits then trying to optimize it is sort of a waste of time. If you are doing it on your own application other than Magento then make sure your application's code is able to serve at least 100 concurrent hits.

Server software, hardware also has a great impact on overall performance, so make it as high as possible. For overcoming common bottlenecks of a web application, we have some predefined rulesets that are described elaborately in Yahoo! and Google for YSlow and Page Speed, respectively. Follow them and you are good to go.

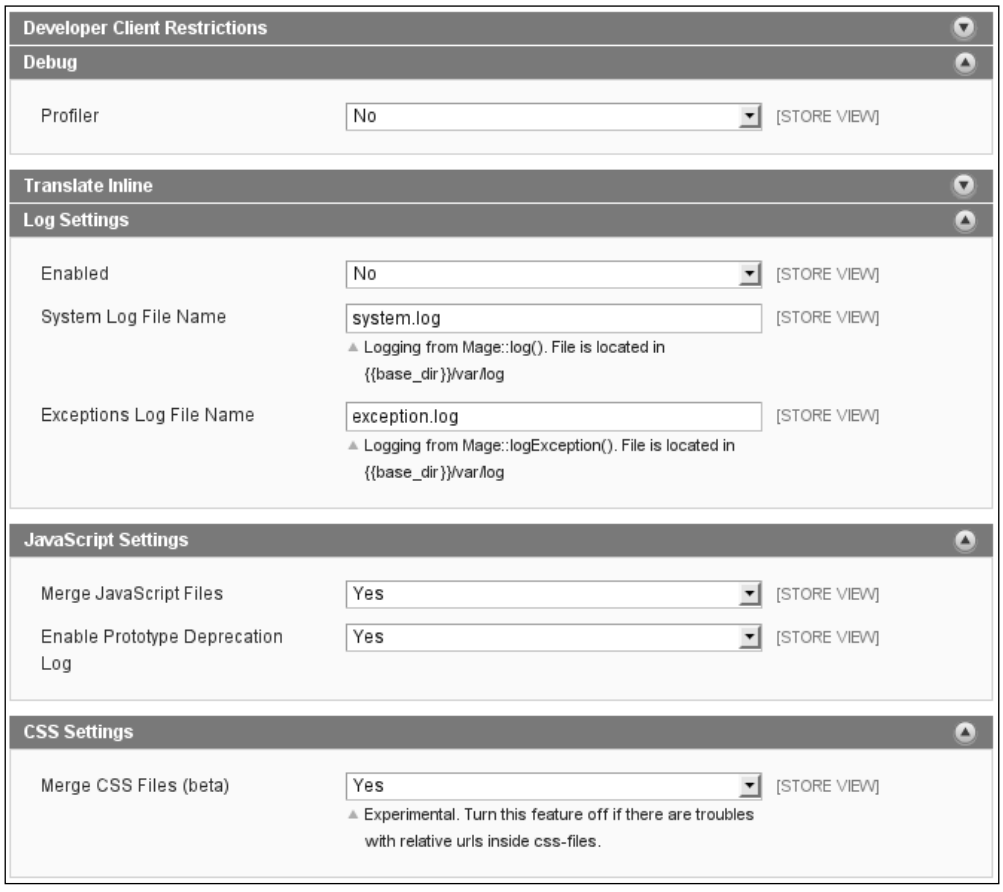
Last but not least, is to host your application in a server located in the same area as your targeted customer's location.

Tuning Magento configurations

Magento configuration is defaulted to run in a wide variety of servers, which might hinder the best performance. This recipe will help you tune the Magento configurations to let it perform fast and furious!

How to do it...

1. Log in to your Magento backend as admin.
2. Cleanup (uninstall and delete) all extensions/modules that you are not using.
3. Enable all types of caching from the Magento backend | **System** | **Cache Management** page. Don't forget to flush/clear all cache storages.
4. Compile your Magento from the **System** | **Tools** | **Compilation** | **Run Compilation** process. It compiles all Magento installation files and creates a single include path. It will speed up pages 25 to 50 percent, according to the official documentation.
5. Go to the page **System** | **Configuration** | **Advanced** | **Disable Modules Output** and disable those which you are not using such as **Mage_poll**.
6. Navigate to the page **System** | **Configurations** | **Advanced** | **Developers** and make it read as follows:



7. Navigate to the **Catalog** | **Attributes** | **Manage Attributes** page. Set only those attribute frontend properties to **Yes** that you're actually going to use. Set all others to **No**. Don't use them in quick search, advanced search compare, and so on.
8. Use store content's mark-up as W3 valid and avoid expressions in CSS like this:

```
background-color: expression( (new Date()).getHours()%2 ?
"#B8D4FF" : "#F08A00" );
```

9. As shown previously, the `expression` method accepts a JavaScript expression. The CSS property is set to the result of evaluating the JavaScript expression. The expression method is ignored by other browsers, so it is useful for setting the properties in Internet Explorer that are needed to create a consistent experience across browsers.

How it works...

These are the configurations provided by the Magento administration panel. We tweaked it to get a better performance. Note that Merging JavaScript and CSS is a beta feature that is available in Magento 1.4.x version. Merging JavaScript and CSS is recommended both by YSlow and Page Speed.

Using APC/Memcached as the cache backend

Magento has got a cache system that is based on files by default. We can boost the overall performance by changing the cache handler to a better engine like APC or Memcached. This recipe will help us to set up APC or Memcached as the cache backend.

Getting ready

Installation of APC:

Alternative PHP Cache (**APC**) is a PECL extension. For any Debian-based Distro, it can be installed with an easy command from the terminal:

```
sudo apt-get install php5-apc
```

Or:

```
sudo pecl install APC
```

You can also install it from the source. The package download location for APC is: `http://pecl.php.net/package/APC`. Check whether it exists or not in `phpinfo()`. If you cannot see an APC block there, then you might not have added APC in the `php.ini` file.

Installation of Memcached:

Memcached is also available in most OS package repositories. You can install it from the command line:

```
sudo apt-get install php5-memcached
```

Memcached could be installed from source as well. Check whether it exists or not in `phpinfo()`. If you cannot see a Memcached block there, then you might not have added Memcached in the `php.ini` file.

You can also check it via the telnet client. Issue the following command in the terminal:

```
telnet localhost 11211
```

We can issue the `get` command now:

```
get greeting
```

Nothing happened? We have to set it first.

```
set greeting 1 0 11
Hello World
STORED
get greeting
Hello World
END
quit
```

How to do it...

1. Okay, we are all set to go for the APC or Memcached. Let's do it now for APC. Open `local.xml` in your favorite PHP editor. Add the cache block as follows:

```
<?xml version="1.0"?>
<config>
  <global>
    <install>
      <date><![CDATA[Sat, 26 Jun 2010 11:55:18 +0000]]></date>
    </install>
    <cache>
      <backend>apc</backend>
      <prefix>alphanumeric</prefix>
    </cache>
    <crypt>
        <key><![CDATA[870f60e1ba58fd34dbf730bfa8c9c152]]></key>
```

```
    </crypt>
    <disable_local_modules>false</disable_local_modules>
    <resources>
      <db>
        <table_prefix><![CDATA[]]></table_prefix>
      </db>
      <default_setup>
        <connection>
          <host><![CDATA[localhost]]></host>
          <username><![CDATA[root]]></username>
          <password><![CDATA[f]]></password>
          <dbname><![CDATA[magento]]></dbname>
          <active>1</active>
        </connection>
      </default_setup>
    </resources>
    <session_save><![CDATA[files]]></session_save>
  </global>
  <admin>
    <routers>
      <adminhtml>
        <args>
          <frontName><![CDATA[backend]]></frontName>
        </args>
      </adminhtml>
    </routers>
  </admin>
</config>
```

2. Delete all files from the `var/cache/` directory. Reload your Magento and benchmark it now to see the boost in performance. Run the benchmark several times to get an accurate result.

   ```
   ab -c 5 -n 100 http://magento.local.com/
   ```

3. You can use either APC or Memcached. Let's test it with Memcached now. Delete the cache block as we set with APC previously and add the cache block as follows:

   ```
   <?xml version="1.0"?>
   <config>
     <global>
       <install>
         <date><![CDATA[Sat, 26 Jun 2010 11:55:18 +0000]]></date>
       </install>
       <crypt>
         <key><![CDATA[870f60e1ba58fd34dbf730bfa8c9c152]]></key>
   ```

```
    </crypt>
    <disable_local_modules>false</disable_local_modules>
    <resources>
      <db>
        <table_prefix><![CDATA[]]></table_prefix>
      </db>
      <default_setup>
        <connection>
          <host><![CDATA[localhost]]></host>
          <username><![CDATA[root]]></username>
          <password><![CDATA[f]]></password>
          <dbname><![CDATA[magento]]></dbname>
          <active>1</active>
        </connection>
      </default_setup>
    </resources>
    <session_save><![CDATA[files]]></session_save>

    <cache>
      <backend>memcached</backend> apc / memcached / xcache /
empty=file
      <slow_backend>file</slow_backend>  database / file (default)
- used for 2 levels cache setup, necessary for all shared memory
storages
      <memcached> memcached cache backend related config
        <servers> any number of server nodes can be included
          <server>
            <host><![CDATA[127.0.0.1]]></host>
            <port><![CDATA[11211]]></port>
            <persistent><![CDATA[1]]></persistent>
            <weight><![CDATA[2]]></weight>
            <timeout><![CDATA[10]]></timeout>
            <retry_interval><![CDATA[10]]></retry_interval>
            <status><![CDATA[1]]></status>
          </server>
        </servers>
        <compression><![CDATA[0]]></compression>
        <cache_dir><![CDATA[]]></cache_dir>
        <hashed_directory_level><![CDATA[]]>
        </hashed_directory_level>
        <hashed_directory_umask><![CDATA[]]>
        </hashed_directory_umask>
        <file_name_prefix><![CDATA[]]></file_name_prefix>
      </memcached>
```

```
    </cache>

  </global>
  <admin>
    <routers>
      <adminhtml>
        <args>
          <frontName><![CDATA[backend]]></frontName>
        </args>
      </adminhtml>
    </routers>
  </admin>
</config>
```

4. Save the `local.xml` file, clear all cache files from `/var/cache/` and reload your Magento in the frontend and check the performance.
5. Mount `var/cache` as TMPFS:

```
mount tmpfs /path/to/your/magento/var/cache -t tmpfs -o size=64m
```

How it works...

Alternative PHP Cache (APC) is a free, open source opcode cache framework that optimizes PHP intermediate code and caches data and compiled code from the PHP bytecode compiler in shared memory, which is similar to Memcached. APC is quickly becoming the de facto standard PHP caching mechanism, as it will be included built-in to the core of PHP, starting with PHP 6. The biggest problem with APC is that you can only access the local APC cache.

Memcached's magic lies in its two-stage hash approach. It behaves as though it were a giant hash table, looking up key = value pairs. Give it a key, and set or get some arbitrary data. When doing a memcached lookup, first the client hashes the key against the whole list of servers. Once it has chosen a server, the client then sends its request, and the server does an internal hash key lookup for the actual item data. Memcached affords us endless possibilities (query caching, content caching, session storage) and great flexibility. It's an excellent option for increasing performance and scalability on any website without requiring a lot of additional resources.

Changing the var/cache to TMPFS is a very good trick to increase disk I/O. I personally found both APC's and Memcached's performance pretty similar. Both are good to go. If you want to split your cache in multiple servers go for the Memcached. Good Luck!

The highlighted sections in code are for the APC and Memcached settings, respectively.

Accelerating PHP: php.ini configuration

If you don't use an APC-like accelerator, then it's time to use one. As, when a PHP script is requested, PHP reads the script and compiles it into what's called Zend opcode, a binary representation of the code to be executed. This opcode is then executed by the PHP engine and thrown away. An opcode cache saves this compiled opcode and reuses it the next time the page is called. This saves a considerable amount of time. In this recipe, we will learn how to optimize the `php.ini` configuration for its best performance.

Getting ready

There are many PHP accelerators out there. You can go for any of them. Such as APC, eAceelerator, XCache, Zend Accelerator (a tool in Zend server), Zend Platform, and so on. Here are some URLs for your convenience:

- APC: `http://pecl.php.net/package/APC`
- EAccelerator: `http://eaccelerator.net/`
- XCache: `http://xcache.lighttpd.net/`

How to do it...

1. The very first job is to use an efficient process manager such as php-fpm, which runs FastCGI with impressive speed.
2. Use `realpath_cache_size` in `php.ini`. This denotes the size of the realpath cache to be used by PHP. This value should be increased on systems where PHP opens many files, to reflect the quantity of the file operations performed. Here is mine:

   ```
   realpath_cache_size=1M
   realpath_cache_ttl=86400
   ```

3. There are some other resources as well, which can improve the overall performance:

Setting	Description	Recommended value
`max_execution_time`	How many CPU-seconds a script can consume	30
`max_input_time`	How long (seconds) a script can wait for input data	60
`memory_limit`	How much memory (bytes) a script can consume before being killed	32M
`output_buffering`	How much data (bytes) to buffer before sending out to the client	4096

4. Last but not the least is to set error reporting to off.

How it works...

These numbers depend mostly on your application. If you accept large files from users, then `max_input_time` may have to be increased, either in `php.ini` or by overriding it in code. Similarly, a CPU- or memory-heavy program may need larger settings. The purpose is to mitigate the effect of a runaway program, so disabling these settings globally isn't recommended. Another note on `max_execution_time`: This refers to the CPU time of the process, not the absolute time. Thus, a program that does lots of I/O and few calculations may run for much longer than `max_execution_time`. It's also how `max_input_time` can be greater than `max_execution_time`.

The amount of logging that PHP can do is configurable. In a production environment, disabling all but the most critical logs saves disk writes. If logs are needed to troubleshoot a problem, you can turn up logging as needed. `error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR` turns on enough logging to spot problems but eliminates a lot of chatter from scripts.

Applying YSlow and Page Speed rules

YSlow and Page Speed both are very useful, FREE tools for optimization experts. Personally, I use both of them. Both have some defined rulesets and options to create on your own by modifying a ruleset. This recipe will let us apply those rules in our application.

Most of the rules have already been applied in our previous recipes in this chapter. We will now deal with the rest of the recipes.

Getting ready

We must have installed the Firefox browser and the following add-ons to work with this recipe. Make sure these tools are installed and regarding versions, the later the better.

Tool	Download link
Firefox	`http://www.mozilla-europe.org/en/firefox/`
Firebug	`http://getfirebug.com/`
YSlow	`http://developer.yahoo.com/yslow/`
Page Speed	`http://code.google.com/speed/page-speed/`

How to do it...

YSlow analyzes a web page on a selected ruleset, Ruleset YSlow(v2) and Ruleset YSlow(v1). v2 comprised 22 rules while v1 has 13 rules. We will be using v2, which has the following rules. Studies have shown that web page response time can be improved by 25 percent to 50 percent by following these rules:

- Minimize HTTP requests
- Use a content delivery network
- Add an expires or a cache-control header
- Gzip components
- Put StyleSheets at the top
- Put scripts at the bottom
- Avoid CSS expressions
- Make JavaScript and CSS external
- Reduce DNS lookups
- Minify JavaScript and CSS
- Avoid redirects
- Remove duplicate scripts
- Configure ETags
- Make AJAX cacheable
- Use GET for AJAX requests
- Reduce the number of DOM elements
- No 404s
- Reduce cookie size
- Use cookie-free domains for components
- Avoid filters
- Do not scale images in HTML
- Make `favicon.ico` small and cacheable

1. If you have not followed the previous recipes of this chapter, you are strongly recommended to follow those and then come back to this one. At this stage, if we run the YSlow test then we should see the overall grade is B. We failed at "use a content delivery network (CDN)". For convenience, I am attaching the full and final `.htaccess` file here:

```
############################################
## uncomment these lines for CGI mode
## make sure to specify the correct cgi php binary file name
## it might be /cgi-bin/php-cgi
```

```
#    Action php5-cgi /cgi-bin/php5-cgi
#    AddHandler php5-cgi .php

###############################################
## GoDaddy specific options

#   Options -MultiViews

## you might also need to add this line to php.ini
##     cgi.fix_pathinfo = 1
## if it still doesn›t work, rename php.ini to php5.ini

###############################################
## this line is specific for 1and1 hosting

    #AddType x-mapp-php5 .php
    #AddHandler x-mapp-php5 .php

###############################################
## default index file

    DirectoryIndex index.php

<IfModule mod_php5.c>

###############################################
## adjust memory limit

#    php_value memory_limit 64M
    php_value memory_limit 128M
    php_value max_execution_time 18000

###############################################
## disable magic quotes for php request vars

    php_flag magic_quotes_gpc off

###############################################
## disable automatic session start
## before autoload was initialized

    php_flag session.auto_start off

###############################################
```

```
## enable resulting html compression

    php_flag zlib.output_compression on

###########################################
# disable user agent verification to not break multiple image
upload

    php_flag suhosin.session.cryptua off

###########################################
# turn off compatibility with PHP4 when dealing with objects

    php_flag zend.ze1_compatibility_mode Off

</IfModule>

<IfModule mod_security.c>
###########################################
# disable POST processing to not break multiple image upload

    SecFilterEngine Off
    SecFilterScanPOST Off
</IfModule>

<IfModule mod_deflate.c>

###########################################
## enable apache served files compression
## http://developer.yahoo.com/performance/rules.html#gzip

    # Insert filter on all content
    SetOutputFilter DEFLATE
    # Insert filter on selected content types only
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
text/css text/javascript

    # Netscape 4.x has some problems...
    BrowserMatch ^Mozilla/4 gzip-only-text/html

    # Netscape 4.06-4.08 have some more problems
    BrowserMatch ^Mozilla/4\.0[678] no-gzip

    # MSIE masquerades as Netscape, but it is fine
```

```
    BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

    # Don›t compress images
    SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-
vary

    # Make sure proxies don›t deliver the wrong content
    Header append Vary User-Agent env=!dont-vary

</IfModule>

<IfModule mod_ssl.c>

##############################################
## make HTTPS env vars available for CGI mode

    SSLOptions StdEnvVars

</IfModule>

<IfModule mod_rewrite.c>

##############################################
## enable rewrites

    Options +FollowSymLinks
    RewriteEngine on

##############################################
## you can put here your magento root folder
## path relative to web root

    #RewriteBase /magento/

##############################################
## workaround for HTTP authorization
## in CGI environment

    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]

##############################################
## always send 404 on missing files in these folders

    RewriteCond %{REQUEST_URI} !^/(media|skin|js)/
```

```
############################################
## never rewrite for existing files, directories and links

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-l

############################################
## rewrite everything else to index.php

    RewriteRule .* index.php [L]

</IfModule>


############################################
## Prevent character encoding issues from server overrides
## If you still have problems, use the second line instead

    AddDefaultCharset Off
    #AddDefaultCharset UTF-8

<IfModule mod_expires.c>

############################################
## Add default Expires header
## http://developer.yahoo.com/performance/rules.html#expires

    ExpiresDefault «access plus 1 year»
    ExpiresActive On
    ExpiresByType image/gif «access plus 30 days»
    ExpiresByType image/jpg «access plus 30 days»
    ExpiresByType image/jpeg «access plus 30 days»
    ExpiresByType image/png «access plus 30 days»
    ExpiresByType image/x-icon «access plus 30 days»
    ExpiresByType text/css «access plus 30 days»
    ExpiresByType application/x-javascript «access plus 30 days»

</IfModule>

############################################
## By default allow all access
```

```
    Order allow,deny
    Allow from all

###############################################
## If running in cluster environment, uncomment this
## http://developer.yahoo.com/performance/rules.html#etags

    FileETag none
```

2. The `.htaccess` is optimized both as recommended by YSlow and Page Speed.
3. Let us optimize our images using a superb tool from Yahoo! named Smush It! At YSlow you will see a link for all tools. Click it and navigate to **All Smush It!**. Upon clicking you would see a page something like the following:

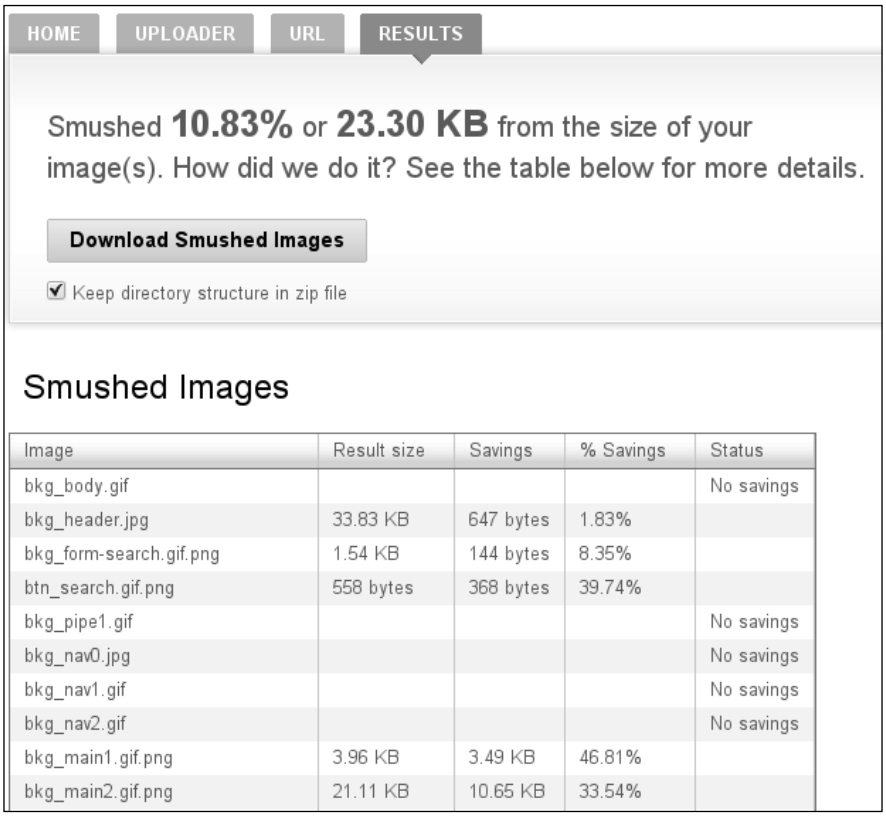
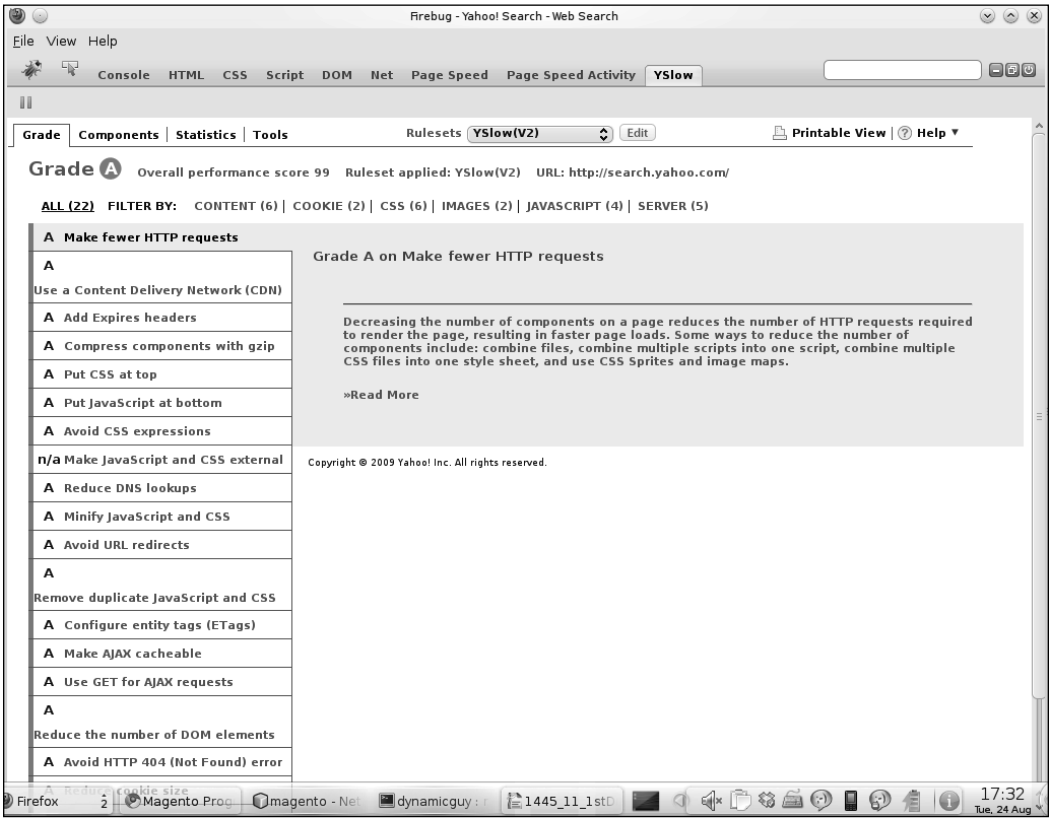


Image	Result size	Savings	% Savings	Status
bkg_body.gif				No savings
bkg_header.jpg	33.83 KB	647 bytes	1.83%	
bkg_form-search.gif.png	1.54 KB	144 bytes	8.35%	
btn_search.gif.png	558 bytes	368 bytes	39.74%	
bkg_pipe1.gif				No savings
bkg_nav0.jpg				No savings
bkg_nav1.gif				No savings
bkg_nav2.gif				No savings
bkg_main1.gif.png	3.96 KB	3.49 KB	46.81%	
bkg_main2.gif.png	21.11 KB	10.65 KB	33.54%	

4. Next, download the optimized images and replace your local images. If you are reading this recipe for some LAMP app other than Magento, then you can use some other tools provided by YSlow such as compression of your js and css files. Kinda cool huh!
5. For Page Speed, it's already applied most of the recommended rules except Parallelize downloads across hostnames. You can adopt it by using a CDN.
6. There are so many CDN providers out there. You can choose any one and pull the static contents from a CDN. After completion of this one, if we run YSlow and Page Speed, it should be all green!



How it works...

YSlow and Page Speed both have online documentation to apply their rulesets. You can and should read it thoroughly to make it better.

- YSlow: `http://developer.yahoo.com/performance/rules.html`.
- Page Speed: `http://code.google.com/speed/page-speed/docs/using.html`.

See also...

You may also refer to the *Measuring/benchmarking your Magento with Siege, ab, Magento profiler, YSlow, Page Speed, GTmetrix, and WebPagetest* recipe of this chapter.

Where to buy this book

You can buy Magento 1.4 Development Cookbook from the Packt Publishing website: `https://www.packtpub.com/magento-1-4-development-cookbook/book`

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

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

www.PacktPub.com

For More Information:
www.packtpub.com/magento-1-4-development-cookbook/book