Chapter 9. Performance Tuning

In this chapter, we will cover the following topics:

* Following best practices
* Speeding up session handling
* Using cache dependencies and chains
* Profiling an application with Yii
* Leveraging HTTP caching
* Combining and minimizing assets
* Running Yii2 on HHVM

Yii is one of the fastest frameworks available. Nevertheless, when developing and deploying an  
application, it is good to have some extra performance for free, and to follow the best practices for the  
application itself. In this chapter, you will see how to configure Yii to gain extra performance. In addition,  
you will learn some best practices for developing an application that will run smoothly until you have very  
high loads.

Following best practices

In this recipe, you will see how to configure Yii2 for the best performance and some additional principles  
of building responsive applications. These principles are both general and Yii-related. Therefore, we will be  
able to apply some of these even without using Yii2.

Getting ready

Create a new yii2-app-basic application using the Composer package manager, as described in the  
official guide at [http://www.yiiframework.com/doc-2.0/gurdestart-installation.html](http://www.yiiframework.com/doc-2.0/guidestart-installation.html).

How to do it...

1. Update your PHP to the latest stable version. Major releases of PHP may bring significant  
   performance improvements. Turn off the debug mode and set the prod environment. This can be  
   done by editing web/index .php as follows:

defined('YII\_DEBUG') or define('YII\_DEBUG', false);  
defined('YII\_ENV') or define('YII\_ENV', 'prod');

Note

Note: In the yii2-app-advanced application skeleton, you can use the shell command php init  
and opt production environment for loading optimized index. php and configuration files.

1. Enable the cache component:

'components' => [

'cache' => [

'class' => 'yii\caching\FileCache',

],

],

You can use any cache storage instead of FileCache. Also, you can register multiple cache  
application components and use Yii: :$app->cache and Yii: :$app->cache2 for different data  
types:

'components' => [

'cache' => [

'class' => 'yii\caching\MemCache',

'useMemcached' => true,

],

'cache2' => [

'class' => 'yii\caching\FileCache',

],

],

The framework uses the cache component by default in its own classes.

1. Enable table schema caching for the db component as follows:

return [

// ...

'components' => [

// ...

'cache' => [

'class' => 'yii\caching\FileCache',

],

'db' => [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=mydatabase',

'username' => 'root',

'password' => '',

'enableSchemaCache' => true,

// Optional. Default value is 3600 seconds  
schemaCacheDuration' => 3600,

// Optional. Default value is 'cache'

'schemaCache' => 'cache',

],

],

];

1. Use plain arrays instead of Active Record objects for listing sets of elements:

$categoriesArray = Categories::find()->asArray()->all();

1. Use each () instead of all() in foreach for a large count of results:

foreach (Post::find()->each() as $post) {

// ...

}

1. Because Composer’s autoloader is used to include most third-party class files, you should consider  
   optimizing it by executing the following command:

composer dump-autoload  
-o

How it works...

When yii\_debug is set to false, Yii turns OFF all the trace level logging and uses less error handling  
code. Also, when you set yii\_env to prod your application does not load Yii and Debug panel modules.

Setting schemaCachingDuration to a number of seconds allows caching the database schema used by  
Yii’s Active Record. This is highly recommended for production servers and it significantly improves the  
Active Record performance. In order for it to work, you need to properly configure the cache component  
as follows:

'cache' => [

'class' => 'yii\cache\FileCache',

],

Enabling the cache also has a positive effect on other Yii components. For example, Yii router or  
urlManager starts to cache routes.

Of course, you can get into a situation where the preceding settings will not help to achieve a sufficient  
performance level. In most cases, it means that either the application itself is a bottleneck or you need  
more hardware.

* Server-side performance is just a part of the big picture: Server-side performance is only one  
  of the things that affect the overall performance. By optimizing the client side such as serving CSS,  
  images, and JavaScript files, proper caching and minimizing the amount of HTTP-requests can give  
  a good visual performance gain even without optimizing the PHP code.
* Things to be done without using Yii: Some things are best done without Yii. For example, image  
  resizing on-the-fly is better in a separate PHP script in order to avoid the extra overhead.
* Active Record versus Query Builder and SQL: Use Query Builder or SQL in performance-  
  critical application parts. Generally, AR is most useful when adding and editing records, as it adds a  
  convenient validation layer, and is less useful when selecting records.
* Always check for slow queries first: Database can become a bottleneck in a second if a  
  developer accidentally forgets to add an index to a table that is being read often or vice versa, or  
  adds too many indexes to a table we are writing to very often. The same goes for selecting  
  unnecessary data and unneeded JOINs.
* Cache or save results of heavy processes: If you can avoid running a heavy process in every  
  page load, it is better to do so. For example, it is a good practice to save or cache results of parsing  
  the markdown text, purify it (this is a very resource-intensive process) once, and then to use the  
  ready-to-display HTML.
* Handling too much processing: Sometimes there is too much processing to be handled  
  immediately. It can be building complex reports or simply sending e-mails (if your project is heavily  
  loaded). In this case, it is better to put it into a queue and process it later using cron or other  
  specialized tools.

See also

For more information about performance tuning and caching refer to the following URLs:

* [http://www.yiiframework.eom/doc-2.0/guide-tutorial-performance-tnning.html](http://www.yiiframework.com/doc-2.0/guide-tutorial-performance-tuning.html)
* [http://www.yiiframework.com/doc-2.0/guide-cachjng-overv'ew.html](http://www.yiiframework.com/doc-2.0/guide-caching-overview.html)