Replication and read-write splitting

In this recipe we will have a look at how to do replication and read-write splitting. We will see how slave  
and master servers help us in getting these done.

Getting ready

1. Create a new application using the Composer package manager, as described in the official guide at  
   <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.
2. Set up the database connection and create a table named post, as follows:

DROP TABLE IF EXISTS 'blog\_post';

CREATE TABLE IF NOT EXISTS 'blog\_post' (

'id' INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

'title' VARCHAR(255) NOT NULL,

'text' TEXT NOT NULL,

'created\_at' INTEGER,

' modified\_at'INTEGER,

PRIMARY KEY ('id')

);

1. Generate the BlogPost model for the table blog\_post.
2. Configure master-slave replication between your database servers, for example, as in the article at  
   [https://www.digitalocean.com/community/tutorials/how-to-set-up-mastpr-slave-replication-in-mysql/](https://www.digitalocean.com/community/tutorials/how-to-set-up-master-slave-replication-in-mysql/).
3. Configure the db component in config/main. php; here’s an example of configuration:

'components' =>

// ..

'db' => [

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

'dsn' => 'mysql:host=4.4.4.4;dbname=masterdb',

'username' => 'master',

'password' => 'pass',

'charset' => 'utf8 ',

'slaveConfig' => [

'username' => 'slave',

'password' => 'pass',

],

// list of slave configurations  
'slaves' => [

['dsn' => 'mysql:host=5.5.5.5;dbname=slavedb']

]

],

// ..

]

How to do it...

1. Create TestController . php as follows:

<?php

namespace app\controllers;

use app\models\BlogPost;  
use Yii;

use yii\helpers\Html;  
use yii\helpers\VarDumper;  
use yii\web\Controller;

/\*\*

* Class TestController
* @package app\controllers  
  \*/

class TestController extends Controller  
{

public function actionIndex(){

$masterModel = new BlogPost();

$masterModel->title = 'Awesome';

$masterModel->text = 'Something is going on..';  
$masterModel->save();

$postId = $masterModel->id;

$replModel = BlogPost::findOne($postId);

return $this->renderContent(

Html::tag('h2', 'Master') .

Html::tag('pre', VarDumper::dumpAsString(

$masterModel

? $masterModel->attributes  
: null

)) .

Html::tag('h2', 'Slave') .

Html::tag('pre', VarDumper::dumpAsString(

$replModel

? $replModel->attributes  
: null

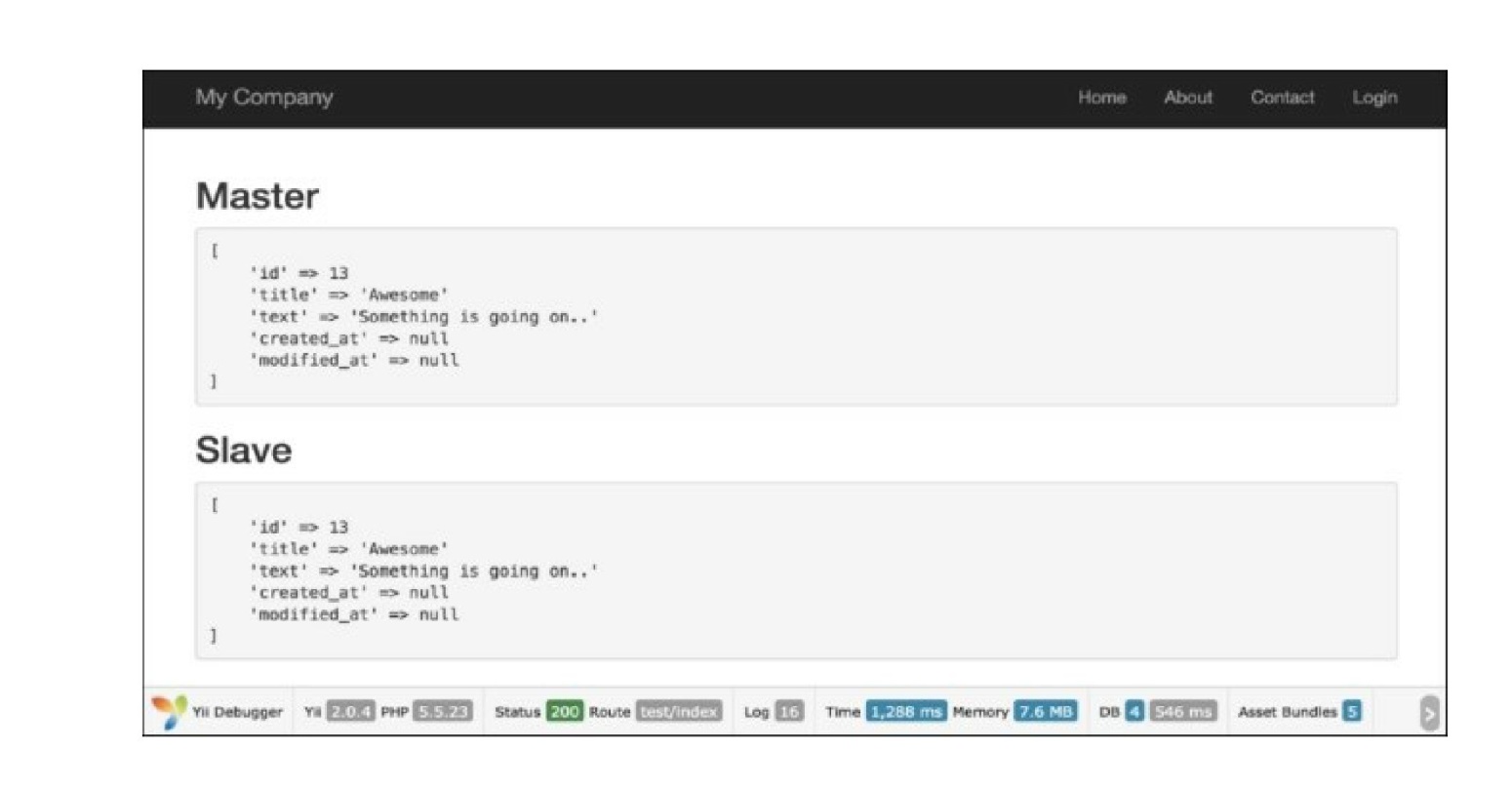
))

);

}

}

2. Run test/index and you should get the output shown in the following screenshot:



How it works...

Slave servers are used for data reading, whereas the master server is used for writing. After the  
ActiveRecord model is saved at the master server, new records, replicate to the slave server and then  
$replModel finds records on it.

There’s more.

The \yii\db\connection component supports load balancing and failover between slaves. When  
performing a read query for the first time, the \yii\db\connection component will randomly pick a slave  
and try connecting to it. If the slave is found dead, it will try another one. If none of the slaves are  
available, it will connect to the master. By configuring a server status cache, a dead server can be  
remembered so that it will not be tried again during a certain period of time.

See also

For further information, refer to the following URLs:

* [http://www.viiframework.com/doc-2.0/guide-db-dao.html#replication-and-read-write-splitting](http://www.yiiframework.com/doc-2.0/guide-db-dao.html%23replication-and-read-write-splitting)
* [http://dev.mysql.com/doc/refman/5.G/en/replication.html](http://dev.mysql.com/doc/refman/5.6/en/replication.html)
* <http://docs.mongodb.org/manual/tutorial/deploy-replica-set/>
* <http://docs.mongodb.org/manual/tutorial/deploy-replica-set-for-testing/>