if (!preg\_match('#A[A-Z]{3}$#s', $source)) {

{

throw new InvalidParamException('Invalid currency format.');

}

}

private function validateDate($date)

{

if (!empty($date) && !preg\_match('#\d{4}\-\d{2}-\d{2}#s', $date)) { throw new InvalidParamException('Invalid date format.');

}

if (empty($date)) {

$date = date('Y-m-d');

}

return $date;

}

private function generateCacheKey($source, $destination, $date)

{

return [ CLASS , $source, $destination, $date];

}

}

4. Attach the component to your config/console .php or config/web. php configuration files:

'components' => [

'cache' => [

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

],

'exchange' => [

'class' => 'app\components\Exchange',

'enableCaching' => true,

],

// ...

db' => $db,

],

5. Right now, we can use a new component directly or via the get method: echo \Yii::$app->exchange->getRate('USD', 'EUR');

echo \Yii::$app->get('exchange')->getRate('USD', 'EUR', '2014-04-12');

6. Create a demonstration console controller:

<?php

namespace app\commands;

use yii\console\Controller;

class ExchangeController extends Controller {

public function actionTest($currency, $date = null)

{

echo \Yii::$app->exchange->getRate('USD', $currency, $date) . PHP\_EOL;

}

}

7. Now try to run any command:

$ ./yii exchange/test EUR > 0.90196 $ ./yii exchange/test EUR 2015-11-24 > 0.93888