$ ./yii exchange/test OTHER

* Exception 'yii\base\InvalidParamException' with message 'Invalid currency format.'

$ ./yii exchange/test EUR 2015/24/11

Exception 'yii\base\InvalidParamException' with message 'Invalid date format.' $ ./yii exchange/test ASD

* Exception 'RuntimeException' with message 'Rate not found.'

As a result, you must see the rate values in the success cases or specific exceptions in the error ones. Besides creating your own components, you can do more.

Overriding existing application components

Most of the time, there will be no need to create your own application components since other types of extension such as widgets or behaviors, cover almost all types of reusable codes. However, overriding core framework components is a common practice and can be used to customize the framework’s behavior for your specific needs without hacking into the core.

For example, to be able to format numbers using the Yii: : app()->formatter->asNumber($value) method instead of our NumberHelper: : format method from the Creating helpers recipe, you can follow the next steps:

1. Extend the yii\i18n\Formatter component as follows:

<?php

namespace app\components;

class Formatter extends \yii\i18n\Formatter {

public function asNumber($value, $decimal = 2)

{

return number\_format($value, $decimal, '.', ',');

}

}

1. Override the class of the built-in formatter component:

'components' => [

// ...

formatter => [

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

],

// ...

],

1. Right now, we can use this method directly:

echo Yii::app()->formatter->asNumber(1534635.2, 3);

Alternatively, it can be used as a new format for the GridView and DetailView widgets: