Our validator class implements the validatevalue() method to support data validation out of the context of a data model. The second method just returns the JavaScript needed for performing client-side validation.

There’s more...

If we would like to hide validator realization, or want to control all validation processes only on the server- side, we can create a Deferred object.

First, modify the Wordsvalidator validator as follows:

<?php

namespace app\components; use yii\validators\Validator; use yii\helpers\Url;

class WordsValidator extends Validator {

public $size = 50;

public $message = 'The number of words must be less than {size}'; public function validateValue($value)

{

if (str\_word\_count($value) > $this->size) {

return ['The number of words must be less than {size}', ['size' => $this-

>size]];

}

return false;

}

public function clientValidateAttribute($model, $attribute, $view)

{

$url = Url::toRoute(['validation/check-words']); return <<<JS

deferred.push($.get("$url", {words: value}).done(function(data) { if (!data.result) {

messages.push(data.error);

}

}));

JS;

}

}

In the preceding code, the deferred variable is provided by Yii, which is an array of Deferred objects. The $.get() jQuery method creates a Deferred object, which is pushed to the deferred array.

Second, add this checkWords action to the validation controller:

public function actionCheckWords()

{

\Yii::$app->response->format = \yii\web\Response::FORMAT\_JSON; $value = Yii::$app->getRequest()->get('words');

$validator = new WordsValidator([

'size' => 10,

]);

$result = $validator->validate($value, $error); return ['result' => $result,'error' => $error ];

}

See also