Creating a custom client-side validation

In the **Writing your own validators** recipe, we created a standalone validator. In this recipe, we will  
modify a validator to create extra client-side validation, which also checks the number of words.

Getting ready

Create a new application by using the Composer package manger, as described in the official guide at  
[http://www.yiiframework.com/doc-2.0/guide-startiustallation.html](http://www.yiiframework.com/doc-2.0/guide-startinstallation.html)**.**

How to do it...

1. Create @app/components/WordsValidator.php as follows:

<?php

namespace app\components;  
use yii\validators\Validator;  
class WordsValidator extends Validator  
{

public $size = 50;

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

{

preg\_match\_all('/(\w+)/i', $value, $matches);  
if (count($matches[0]) > $this->size) {

return [$this->message, ['size' => $this->size]];

}

}

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

{

$message = strtr($this->message, ['{size}' => $this->size]);  
return <<<JS

if (value.split(/\w+/gi).length > $this->size ) {  
messages.push("$message");

}

JS;

}

}

1. Create @app/models/Article . php as follows:

<?php

namespace app\models;  
use app\components\WordsValidator;  
use yii\base\Model;  
class Article extends Model  
{

public $title;

public function rules()

{

return [

['title ', 'string'],

['title', WordsValidator::className(), 'size' => 10],

];

}

}

1. Create @app/controllers/ValidationController . php as follows:

<?php

namespace app\controllers;  
use app\models\Article;  
use Yii;

use yii\web\Controller;

class ValidationController extends Controller  
{

public function actionIndex()

{

$model = new Article();

if ($model->load(Yii::$app->request->post()) && $model->validate()) {  
Yii::$app->session->setFlash('success', 'Model is valid');

}

return $this->render('index', [

'model' => $model,

]);

}

}

4. Create @app/views/validation/index . php as follows:

<?php

use yii\bootstrap\ActiveForm;  
use yii\helpers\Html;

?>

<h1>Article form</h1>

<?php if (Yii::$app->session->hasFlash('success')): ?>

<div class="alert alert-success"><?= Yii::$app->session->getFlash('success'); ?  
></div>

<?php endif; ?>

<?php $form = ActiveForm::begin(); ?>

<?= $form->field($model, 'title') ?>

<div class="form-group">

<?= Html::submitButton('Submit', ['class' => 'btn btn-primary']) ?>

</div>

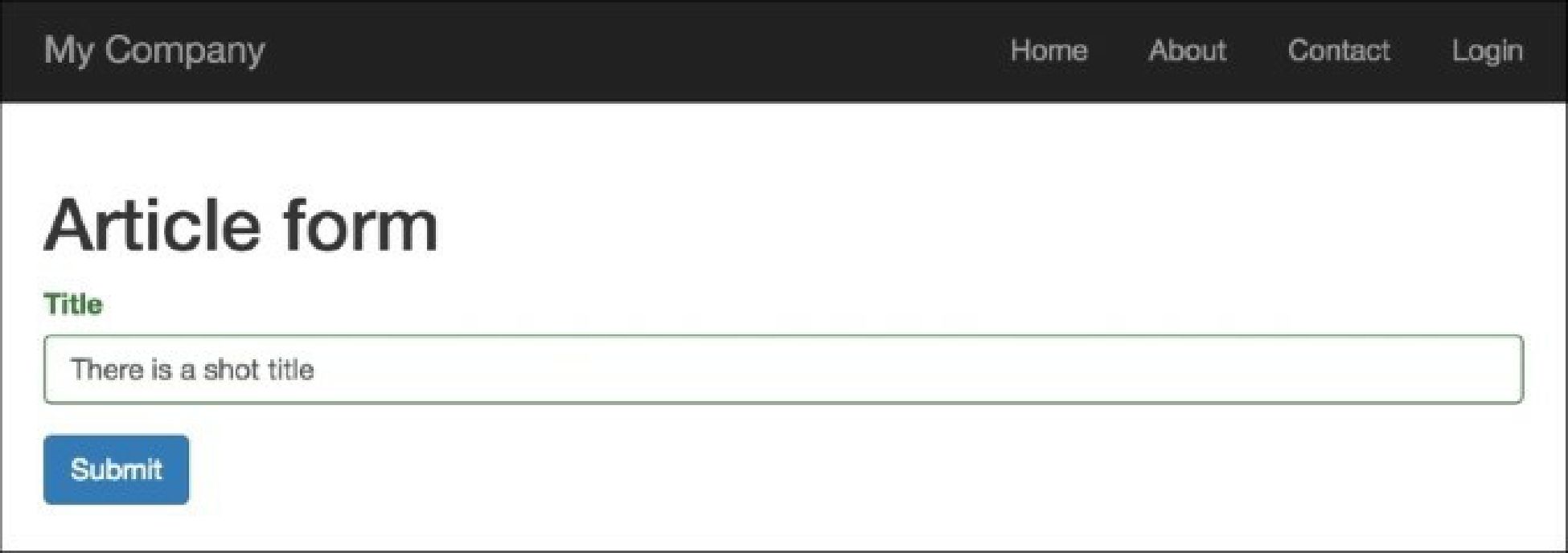
<?php ActiveForm::end(); ?>

How it works...

Run the validation controller by opening index. php?r=validation. You will see an example of an  
incorrect value if you enter more than ten words:



If you enter fewer than ten words, client-side validation will be successful:



First, we created @app/components/wordsvalidator. php, which extends the

@yii\validators\validator class, and added the newly-created validator class to the title attribute of the  
Article model:

['title', WordsValidator::className(), 'size' => 10],

Inside our validator, we’ve defined two special methods: validatevalue() and  
clientValidateAttribute().

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

For further information, refer to the following URLs:

* [http://www.yiiframework.com/doc-2.0/guide-input-validation.html#implementing-client-side-](http://www.yiiframework.com/doc-2.0/guide-input-validation.html%23implementing-client-side-validation) **validation**
* [http://www.viiframework.com/doc-2.0/guide-input-validation.hrml#deferred-validation](http://www.yiiframework.com/doc-2.0/guide-input-validation.html%23deferred-validation)