Writing your own validators

Yii provides a good set of built-in form validators that cover the most typical developer needs and are  
highly configurable. However, in some cases, a developer may need to create a custom validator.

This recipe is a good example of creating a standalone validator that checks the number of words.

Getting ready

Create a new application by using the Composer package manager, as described in the official guide at  
<http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.

How to do it...

1. Create a standalone validator at @app/components/WordsValidator. php as follows:

<?php

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

public $size = 50;

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;

}

}

1. Create an Article model at @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/ModelValidationController . php as follows:

<?php

namespace app\controllers;  
use app\models\Article;  
use yii\helpers\Html;  
use yii\web\Controller;

class ModelValidationController extends Controller  
{

private function getLongTitle()

{

return 'There is a very long content for current article, '.'it should be  
less then ten words';

}

private function getShortTitle()

{

return 'There is a shot title';

}

private function renderContentByModel($title)

{

$model = new Article();

$model->title = $title;  
if ($model->validate()) {

$content = Html::tag('div', 'Model is valid.',[

'class' => 'alert alert-success',

]);

} else {

$content = Html::errorSummary($model, [

'class' => 'alert alert-danger',

]);

}

return $this->renderContent($content);

}

public function actionSuccess()

{

$title = $this->getShortTitle();

return $this->renderContentByModel($title);

}

public function actionFailure()

{

$title = $this->getLongTitle();

return $this->renderContentByModel($title);

}

}

4. Run the success action of the modelValidation controller by opening the index.php?r=model-  
validation/success URL, and you’ll get the following:



5. Run the failure action of the modelValidation controller by opening the index.php?r=model-  
validation/failure URL, and you’ll get the following:



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

<?php

namespace app\controllers;

use app\components\WordsValidator;

use app\models\Article;

use yii\helpers\Html;

use yii\web\Controller;

class AdhocValidationController extends Controller  
{

private function getLongTitle()

{

return 'There is a very long content for current article, '.'it should be  
less then ten words';

}

private function getShortTitle()

{

return 'There is a shot title';

}

private function renderContentByTitle($title)

{

$validator = new WordsValidator([

'size' => 10,

]);

if ($validator->validate($title, $error)) {

$content = Html::tag('div', 'Value is valid.',[

'class' => 'alert alert-success',

]);

} else {

$content = Html::tag('div', $error, [

'class' => 'alert alert-danger',

]);

}

return $this->renderContent($content);

}

public function actionSuccess()

{

$title = $this->getShortTitle();

return $this->renderContentByTitle($title);

}

public function actionFailure()

{

$title = $this->getLongTitle();

return $this->renderContentByTitle($title);

}

}

1. Run the success action of the AdhocValidationController by opening the index.php?r=adhoc-

validation/success URL, and you’ll get the following:



8. Run the failure action of the adhocvalidation controller by opening the index.php?r=adhoc-  
validation/f ailure URL, and you’ll get the following:



How it works...

First, we created a standalone validator that checks the number of words by using the standard  
str\_word\_count PHP function, and then demonstrated two validator use cases:

* Using the validator as a validation rule in the Article model
* Using the validator as an ad hoc validator

The validator has a size attribute, which sets the maximum value for the number of words.

See also

For further information, refer to the following URLs:

* <http://www.yiiframework.com/doc-2.0/guide-input-validation.html>
* <http://www.yiiframework.com/doc-2.0/guide-tutorial-corevalidators.html>