Creating a custom filter

Filters are objects that run before and/or after controller actions. For example, an access control filter may run before actions to ensure that they are allowed to be accessed by particular end users; a content compression filter may run after actions to compress the response content before sending them out to end users.

A filter may consist of a prefilter (filtering logic applied before actions) and/or a postfilter (logic applied after actions). Filters are essentially a special kind of behavior. Therefore, using filters is the same as using behaviors.

Let’s assume that we have a web application, which provides a user interface for working only at specified hours, for example, from 10 AM to 6 PM.

Getting ready

Create a new application using the Composer package manager, as described in the official guide at [http://www.y]'iframework.com/doc-2.0/guide-start-installation.html](http://www.yiiframework.com/doc-2.0/guide-start-installation.html).

How to do it...

1. Create a controller, @app/controllers/TestController. php, as follows:

<?php

namespace app\controllers;

use app\components\CustomFilter; use yii\helpers\Html; use yii\web\Controller;

class TestController extends Controller {

public function behaviors()

{

return [

'access' => [

'class' => CustomFilter::className(),

],

];

}

public function actionIndex()

{

return $this->renderContent(Html::tag('h1',

'This is a test content'

));

}

}

1. Create a new filter, @app/components/CustomFilter . php, as follows:

<?php

namespace app\components;

use Yii;

use yii\base\ActionFilter; use yii\web\HttpException;

class CustomFilter extends ActionFilter {

const WORK\_TIME\_BEGIN = 10; const WORK\_TIME\_END = 18;

protected function canBeDisplayed()

{

$hours = date('G');

return $hours >= self::WORK\_TIME\_BEGIN && $hours <= self::WORK\_TIME\_END;

}

public function beforeAction($action)

{

if (!$this->canBeDisplayed())

{

$error = 'This part of website works from '

. self::WORK\_TIME\_BEGIN . ' to '

. self::WORK\_TIME\_END . ' hours.';

throw new HttpException(403, $error);

}

return parent::beforeAction($action);

}

public function afterAction($action, $result)

{

if (Yii::$app->request->url == '/test/index') {

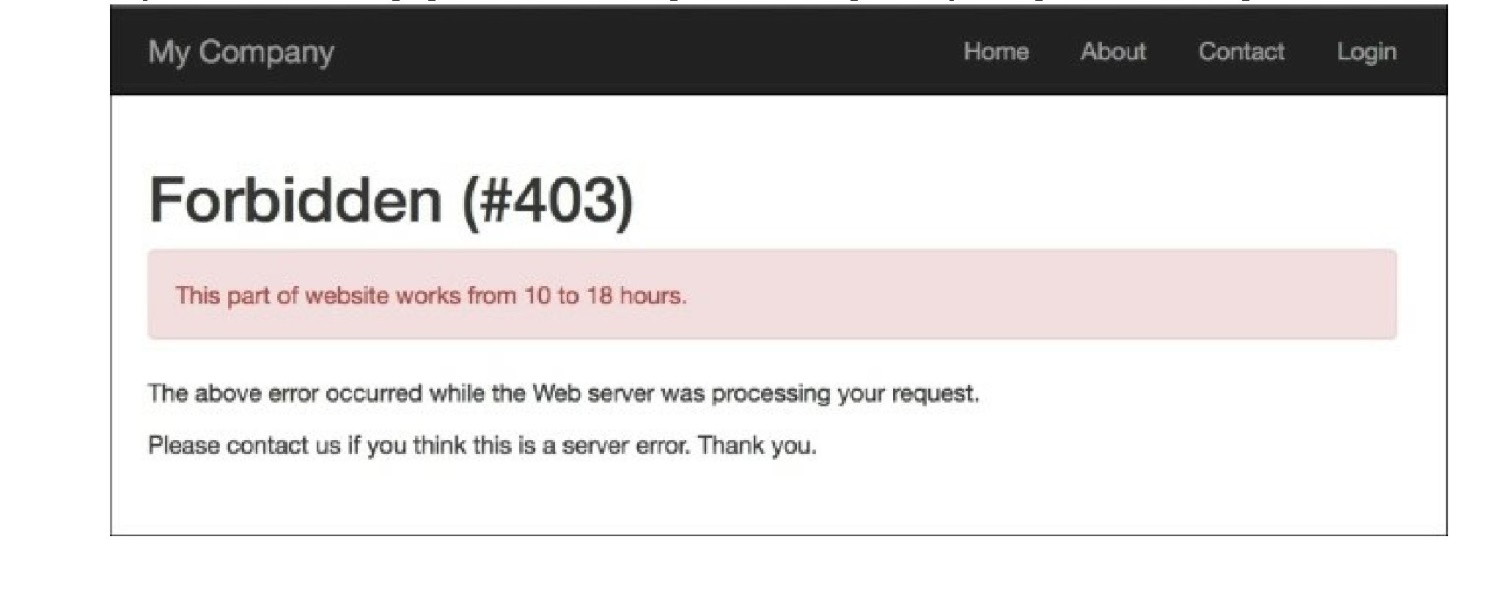
Yii::trace("This is the index action");

}

return parent::afterAction($action, $result);

}

}

3. If you’ve visited this page outside of the specified time period, you’ll get the following:

How it works...

At first, we added a piece of code to our controller, which implements our custom filter:

public function behaviors()

{

return [

'access' => [

'class' => CustomFilter::className(),

],

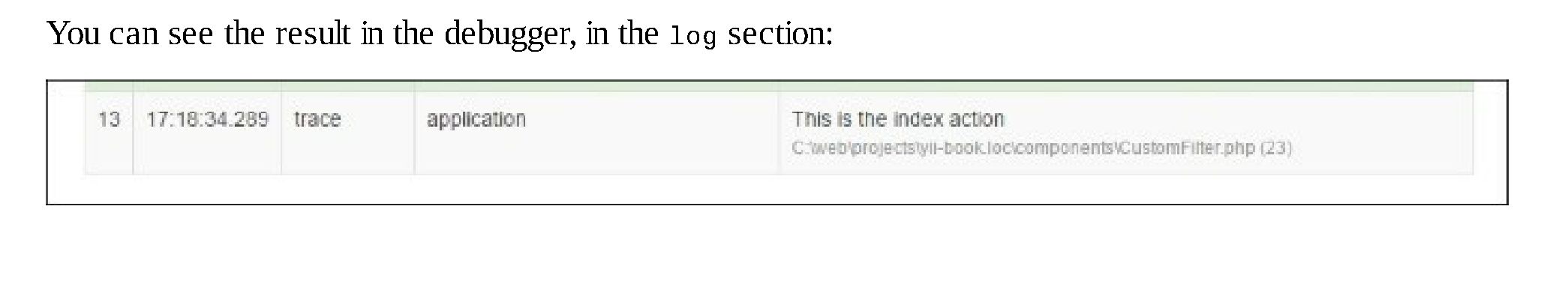
];

}

By default, the filter applies to all actions of the controller, but we can specify actions for which it will be applied, or even exclude actions from our filter.

You have two actions inside it—beforeAction and afterActions. The first one runs before the controller’s actions and the next one after.

In our simple example, we defined a condition which doesn’t allow access to website if the time is earlier than 10 AM, and in the after method we just run a trace method if the current path is test/index.

In real applications, filters are more complex and also, Yii2 provides a lot of built-in filters, such as core, authentication, content negotiator, HTTP cache end, and so on.

See also

For further information, refer to [http://www.yiiframework.com/doc-2.0/guidestructnre-filtprs.html](http://www.yiiframework.com/doc-2.0/guidestructure-filters.html).