Creating filters

A filter is a class that can run before/after an action is executed. It can be used to modify execution  
context or decorate output. In our example, we’ll implement a simple access filter that will allow the user  
to see private content only after accepting the User agreement.

Getting ready

Create a new yii2-app-basic application using the composer, as described in the official guide at  
<http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.

How to do it...

1. Create the agreement form model:

<?php

namespace app\models;

use yii\base\Model;

class AgreementForm extends Model  
{

public $accept;

public function rules()

{

return [

['accept', 'required'],

['accept', 'compare', 'compareValue' => 1, 'message' => 'You must agree  
the rules.'],

];

}

public function attributeLabels()

{

return [

'accept' => 'I completely accept the rules.'

];

}

}

1. Create the agreement checker service:

<?php

namespace app\services;  
use Yii;

use yii\web\Cookie;

class AgreementChecker  
{

public function isAllowed()

{

return Yii::$app->request->cookies->has('agree');

}

public function allowAccess()

{

Yii::$app->response->cookies->add(new Cookie([

'name' => 'agree',

'value' => 'on',

'expire' => time() + 3600 \* 24 \* 90, // 90 days

]));

}

}

1. It encapsulates work with the agreement cookies.

1. Create the filter class:

<?php

namespace app\filters;

use app\services\AgreementChecker;  
use Yii;

use yii\base\ActionFilter;

class AgreementFilter extends ActionFilter  
{

public function beforeAction($action)

{

$checker = new AgreementChecker();  
if (!$checker->isAllowed()) {

Yii::$app->response->redirect(['/content/agreement'])->send();  
return false;

}

return true;

}

}

1. Create the content controller and attach the filter to its behaviors:

<?php

namespace app\controllers;

use app\filters\AgreementFilter;  
use app\models\AgreementForm;  
use app\services\AgreementChecker;  
use Yii;

use yii\web\Controller;

class ContentController extends Controller  
{

public function behaviors()

{

return [

[

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

'only' => ['index'],

],

];

}

public function actionIndex()

{

return $this->render('index');

}

public function actionAgreement()

{

$model = new AgreementForm();

if ($model->load(Yii::$app->request->post()) && $model->validate()) {

$checker = new AgreementChecker();

$checker->allowAccess();

return $this->redirect(['index']);

} else {

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

'model' => $model,

]);

}

}

}

1. Add the views/content/index. php view with private content:

<?php

use yii\helpers\Html;

/\* @var $this yii\web\View \*/

$this->title = 'Content';

$this->params['breadcrumbs'][] = $this->title;

?>

<div class="site-about">

<h1><?= Html::encode($this->title) ?></h1>

<div class="well">

This is our private page.

</div>

</div>

1. Add the views/content/agreement.php view with the form:

<?php

use yii\helpers\Html;

use yii\bootstrap\ActiveForm;

/\* @var $this yii\web\View \*/

/\* @var $form yii\bootstrap\ActiveForm \*/

/\* @var $model app\models\AgreementForm \*/

$this->title = 'User agreement';

$this->params['breadcrumbs'][] = $this->title;

?>

<div class="site-login">

<h1><?= Html::encode($this->title) ?></h1>

<p>Please agree with our rules:</p>

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

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

<div class="form-group">

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

<?= Html::a('Cancel', ['/site/index'], ['class' => 'btn btn-danger']) ?>  
</div>

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

</div>

1. Add the main menu item to the views/layouts/main.php file:  
   echo Nav::widget([

'options' => ['class' => 'navbar-nav navbar-right'],

'items' => [

['label' => 'Home', 'url' => ['/site/index']],

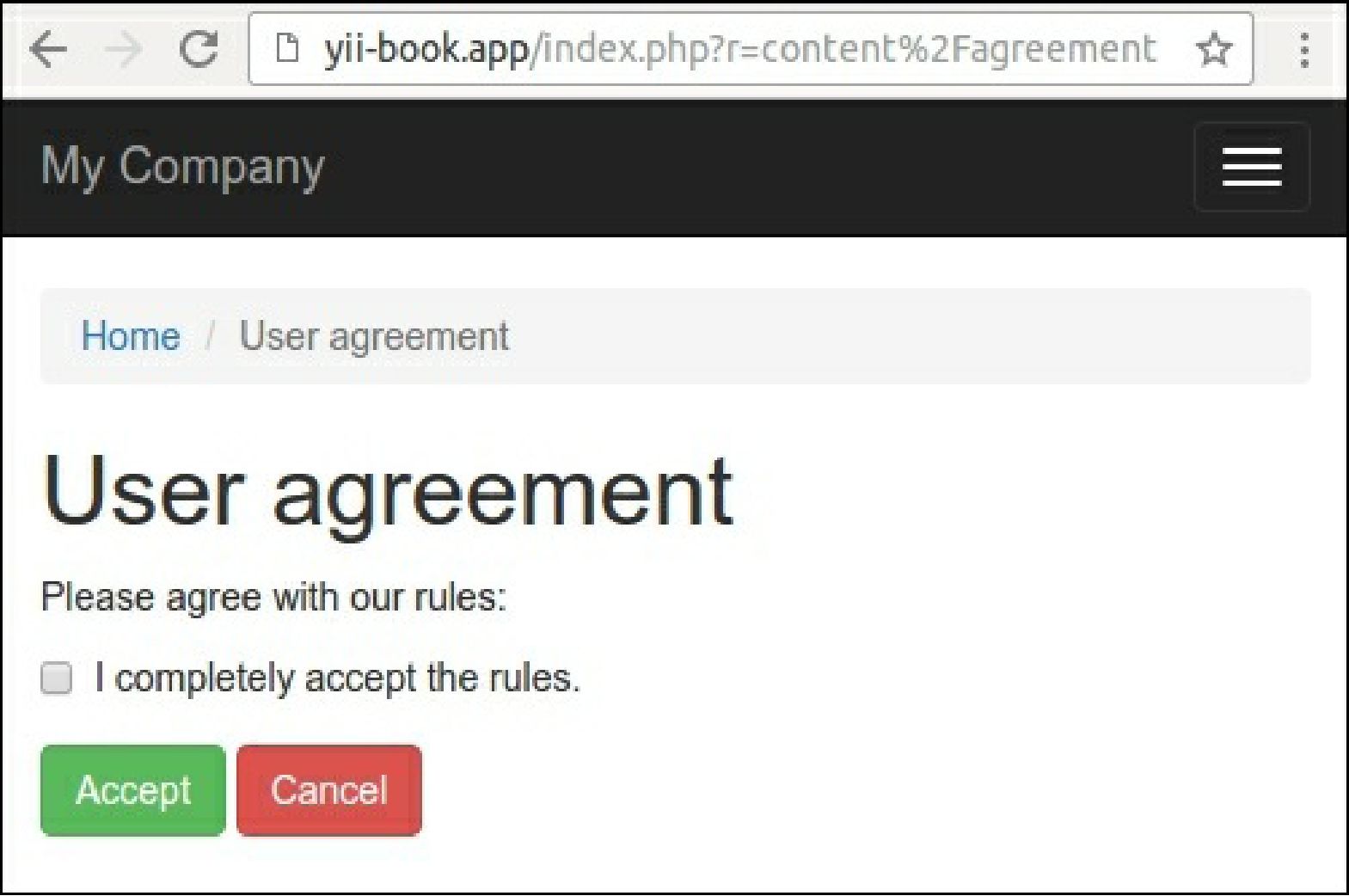
['label' => 'Content', 'url' => ['/content/index']],

['label' => 'About', 'url' => ['/site/about']],

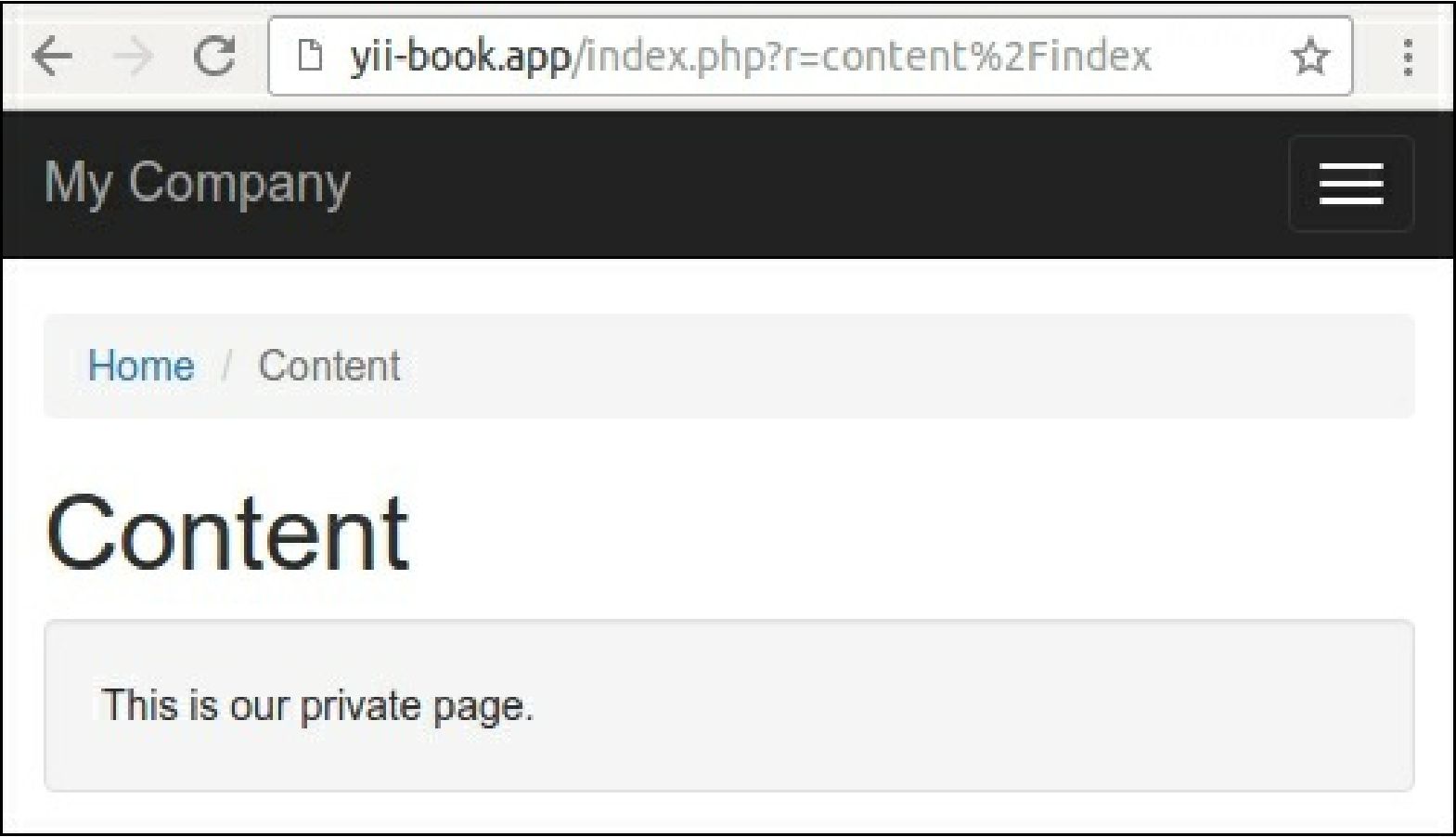
],

]);

8. Try to open the content page. The filter must redirect you to the agreement page:



9. Only after accepting the rules can you see the private content:



10. Also, you can attach the filter to other controllers or modules.

How it works...

A filter should extend the yii\base\ActionFilter class, which extends yii\base\Behavior. We can  
override the beforeAction or afterAction method if we want to do post- and pre-filtering.

For example, we can check user access and throw corresponding HTTP-exceptions in a fail case. In this  
recipe, we redirect the user to the agreement page if the specific cookie value does not exist:

class AgreementFilter extends ActionFilter  
{

public function beforeAction($action)

{

$checker = new AgreementChecker();  
if (!$checker->isAllowed()) {

Yii::$app->response->redirect(['/content/agreement'])->send();  
return false;

}

return true;

}

}

You can attach filters to any controller or module. To specify the list of necessary routes, just use the only  
or except options. For example, we apply our filter only for the index action of the controller:

public function behaviors()

{

return [

[

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

' only' => ['index'],

],

];

}

Note

Do not forget to return a true value in the success case from the beforeAction method. Otherwise, the  
controller action will not be executed.

See also

[For more information about filters, refer to http://www.yiiframework. c om/doc -2.0/guide-structure -](http://www.yiiframework.com/doc-2.0/guide-structure-filters.html)  
filters.html.

For build-in cache and access control filters, refer to:

* <http://www.yiiframework.com/doc-2.0/guide-caching-http.html>
* [http://www.yiiframework.com/doc-2.0/guide-securityauthorizat.ion.ht.ml](http://www.yiiframework.com/doc-2.0/guide-securityauthorization.html)
* The Creating model behaviors recipe