Creating a custom view renderer

There are many PHP template engines out there. Yii2 only offers native PHP templates. If you want to  
use one of the existing template engines or create your own one, you have to implement it—of course, if  
it’s not yet implemented by the Yii community.

In this recipe we’ll re-implement the Smarty templates support.

Подготовка

1. Создайте новое yii2-app-basic приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
   <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
   По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation>
2. Install the Smarty library:

composer require smarty/smarty

How to do it...

Carry out the following steps for creating a custom view renderer:

1. Create the smarty/ViewRenderer . php file:

<?php

namespace app\smarty;

use Smarty;  
use Yii;

class ViewRenderer extends \yii\base\ViewRenderer  
{

public $cachePath = '@runtime/smarty/cache';  
public $compilePath = '@runtime/smarty/compile';

/\*\*

\* @var Smarty  
\*/

private $smarty;

public function init()

{

$this->smarty = new Smarty();

$this->smarty->setCompileDir(Yii::getAlias($this->compilePath));  
$this->smarty->setCacheDir(Yii::getAlias($this->cachePath));  
$this->smarty->setTemplateDir([

dirname(Yii::$app->getView()->getViewFile()),

Yii::$app->getViewPath(),

]);

}

public function render($view, $file, $params)

{

$templateParams = empty($params) ? null : $params;

$template = $this->smarty->createTemplate($file, null, null,  
$templateParams, false);

$template->assign('app', \Yii::$app);

$template->assign('this', $view);

return $template->fetch();

}

}

1. Now we need to connect the view Tenderer to the application. In config/web php, we need to add  
   Tenderers of the view component:

'components' => [

'view' => [

'renderers ' => [

'tpl' => [

'class' => 'app\smarty\ViewRenderer',

],

],

].

];

1. Now let’s test it. Create a new SmartyController:

<?php

namespace app\controllers;  
use yii\web\Controller;

class SmartyController extends Controller  
{

public function actionIndex()

{

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

'name' => 'Bond ',

]);

}

}

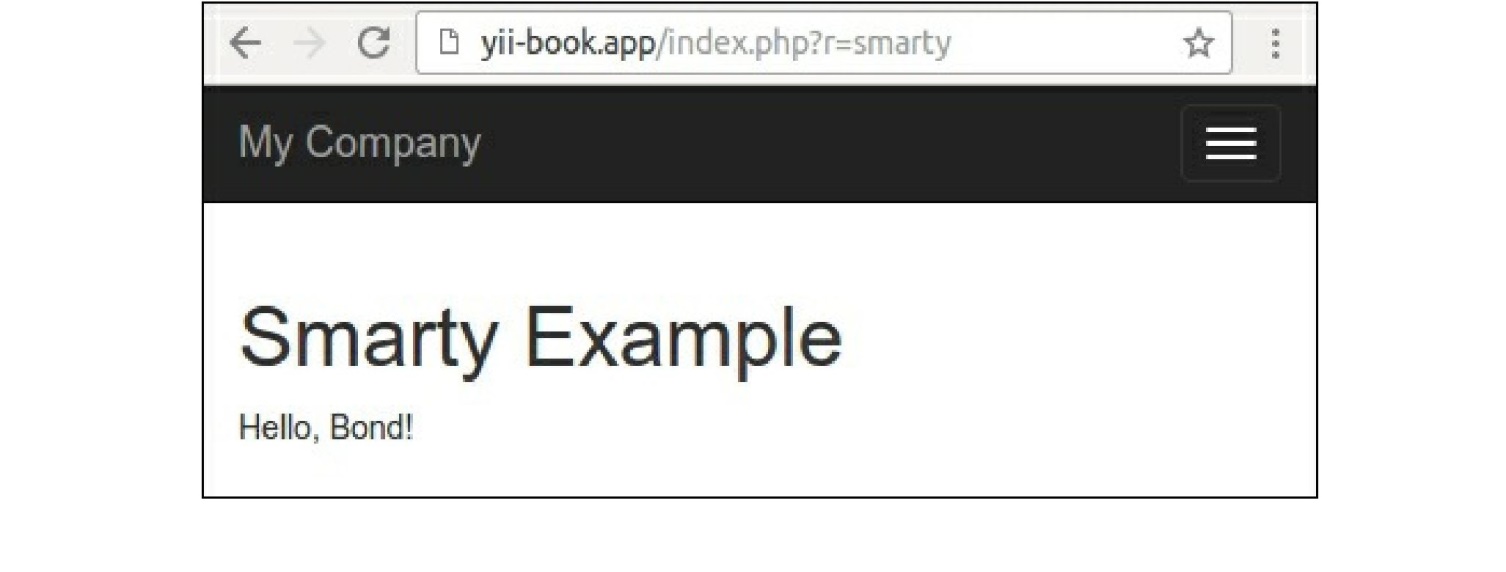
1. Next, we need to create the views/smarty/index. tpl view:

<div class="smarty-index">

<h1>Smarty Example</h1>

<p>Hello, {$name}!</p>

</div>

1. Now try running the controller. In a success case, you should get the following as output:

How it works...

A view Tenderer is a child of the yii\base\viewRenderer abstract class that implements only one method,  
called render:

<?php

namespace yii\base;

abstract class ViewRenderer extends Component  
{

/\*\*

* Renders a view file.

\*

* This method is invoked by [[View]] whenever it tries to render a view.
* Child classes must implement this method to render the given view file.

\*

* @param View $view the view object used for rendering the file.
* @param string $file the view file.
* @param array $params the parameters to be passed to the view file.
* @return string the rendering result  
  \*/

abstract public function render($view, $file, $params);

}

Therefore, we are getting a view component, file path, and render variables. We need to process the file  
and return the rendered result. In our case, processing itself is done by the Smarty template engine, so we  
need to properly initialize it and call its processing methods:

class ViewRenderer extends \yii\base\ViewRenderer  
{

public $cachePath = '@runtime/smarty/cache';  
public $compilePath = '@runtime/smarty/compile';  
private $smarty;

public function init()

{

$this->smarty = new Smarty();

$this->smarty->setCompileDir(Yii::getAlias($this->compilePath));  
$this->smarty->setCacheDir(Yii::getAlias($this->cachePath));  
$this->smarty->setTemplateDir([

dirname(Yii::$app->getView()->getViewFile()),

Yii::$app->getViewPath(),

]);

}

}

It is a good practice to store Yii temporary files in the application runtime directory. That is why we are  
setting the compile directory, where Smarty stores its templates compiled into PHP, to  
runtime/smarty/compile.

Rendering itself is a bit simpler:

public function render($view, $file, $params)

{

$templateParams = empty($params) ? null : $params;

$template = $this->smarty->createTemplate($file, null, null, $templateParams, false);  
$template->assign('app', \Yii::$app);

$template->assign('this', $view);  
return $template->fetch();

}

All data set via $this->render is passed to the Smarty template as it is. Also, we are creating special  
Smarty template variables named app and this that point to Yii: :$app and Yii: :$app->view and allow  
us to get application properties inside a template.

Then, we are rendering the templates.

See also

You can get ready to use Smarty view renderer with plugins and configuration support at  
[https ://githnb.com/yiisoft/yii2-smarty](https://github.com/yiisoft/yii2-smarty).

To learn more about Smarty and view renderers in general, refer to the following URLs:

* [http://www. smarty.net](http://www.smarty.net)
* [http://www.viiframework.com/doc-2.0/guide-tutorial-templateengines.html](http://www.yiiframework.com/doc-2.0/guide-tutorial-templateengines.html)
* <http://www.yiiframework.com/doc-2.0/guide-structure-views.html>