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;