

3. Therefore, the basic rule is to always escape all dynamic data. For example, we should do the same for a link name:

use \yii\helpers\Html;

echo Html::a(Html::encode($\_GET['username']), array());

That’s it. You have a page that is free from XSS. Now, what if we want to allow some HTML to pass? We cannot use \yii\helpers\Html: : encode anymore because it will render HTML as just a code and we need the actual representation. Fortunately, there is a tool bundled with Yii that allows you to filter the malicious HTML. It is named html Purifier and can be used in the following way:

<?php

namespace app\controllers; use Yii;

use yii\helpers\Html;

use yii\helpers\HtmlPurifier;

use yii\web\Controller;

/\*\*

\* Class SiteController.

\* @package app\controllers \*/

class XssController extends Controller {

/\*\*

\* @return string \*/

public function actionIndex()

{

$username = Yii::$app->request->get('username', 'nobody');

$content = Html::tag('h1', 'Hello, ' . $username . '!');

return $this->renderContent(