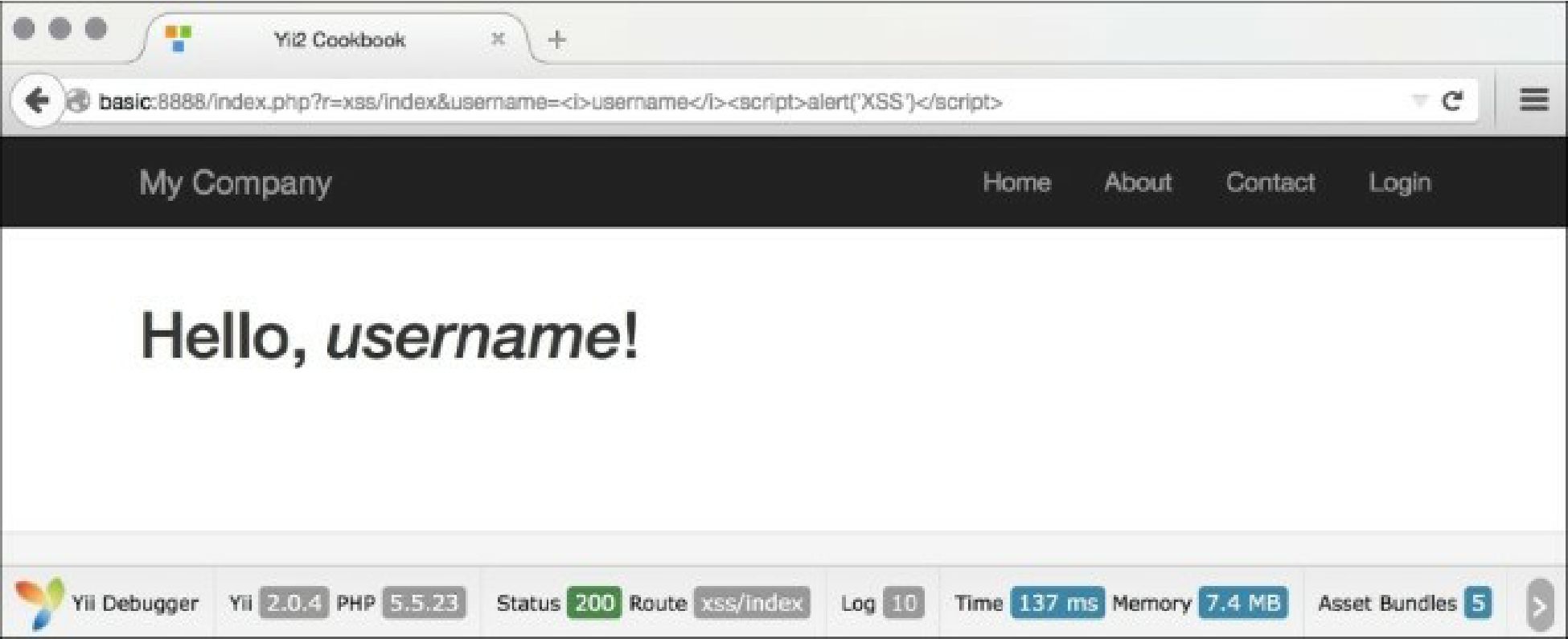
HtmlPurifier::process($content)

}

);

}

Now if we access the HTML action using a URL such as /xss/index?username=<i>username</i> ! <script>alert( 'xss' )</script>, HTML Purifier will remove the malicious part and we will get the following result:



How it works...

1. Internally, \yii\helpers\Html: :encode looks like the following:

public static function encode($content, $doubleEncode = true)

{

return htmlspecialchars($content, ENT\_QUOTES | ENT\_SUBSTITUTE, Yii::$app ? Yii::$app->charset : 'UTF-8', $doubleEncode);

}

1. So basically, we use PHP’s internal htmlspecialchars function, which is pretty secure if one does not forget to pass the correct charset in the third argument.

\yii\helpers\HtmlPurifier uses the HTML Purifier library, which is the most advanced solution out there to prevent XSS inside of HTML. We have used its default configuration, which is okay for most user-entered content.

There’s more.

There are more things to know about XSS and HTML Purifier; they are discussed in the following section.