Filter.ExtractStyleBlocks

TYPE: bool

VERSION: 3.1.0

DEFAULT: false

EXTERNAL: CSSTidy

--DESCRIPTION--

<p>

This directive turns on the style block extraction filter, which removes

<code>style</code> blocks from input HTML, cleans them up with CSSTidy,

and places them in the <code>StyleBlocks</code> context variable, for further

use by you, usually to be placed in an external stylesheet, or a

<code>style</code> block in the <code>head</code> of your document.

</p>

<p>

Sample usage:

</p>

<pre><![CDATA[

<?php

header('Content-type: text/html; charset=utf-8');

echo '<?xml version="1.0" encoding="UTF-8"?>';

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>

<title>Filter.ExtractStyleBlocks</title>

<?php

require\_once '/path/to/library/HTMLPurifier.auto.php';

require\_once '/path/to/csstidy.class.php';

$dirty = '<style>body {color:#F00;}</style> Some text';

$config = HTMLPurifier\_Config::createDefault();

$config->set('Filter', 'ExtractStyleBlocks', true);

$purifier = new HTMLPurifier($config);

$html = $purifier->purify($dirty);

// This implementation writes the stylesheets to the styles/ directory.

// You can also echo the styles inside the document, but it's a bit

// more difficult to make sure they get interpreted properly by

// browsers; try the usual CSS armoring techniques.

$styles = $purifier->context->get('StyleBlocks');

$dir = 'styles/';

if (!is\_dir($dir)) mkdir($dir);

$hash = sha1($\_GET['html']);

foreach ($styles as $i => $style) {

file\_put\_contents($name = $dir . $hash . "\_$i");

echo '<link rel="stylesheet" type="text/css" href="'.$name.'" />';

}

?>

</head>

<body>

<div>

<?php echo $html; ?>

</div>

</b]]><![CDATA[ody>

</html>

]]></pre>

<p>

<strong>Warning:</strong> It is possible for a user to mount an

imagecrash attack using this CSS. Counter-measures are difficult;

it is not simply enough to limit the range of CSS lengths (using

relative lengths with many nesting levels allows for large values

to be attained without actually specifying them in the stylesheet),

and the flexible nature of selectors makes it difficult to selectively

disable lengths on image tags (HTML Purifier, however, does disable

CSS width and height in inline styling). There are probably two effective

counter measures: an explicit width and height set to auto in all

images in your document (unlikely) or the disabling of width and

height (somewhat reasonable). Whether or not these measures should be

used is left to the reader.

</p>

--# vim: et sw=4 sts=4