htmlspecialchars 関数を使うタイミング v1.0

Seiichi Nukayama

2022 年 2 月 14 日

目次

1　htmlspecialchars 関数の働き

htmlspecialchars 関数の働きは、以下のようなものである。

```
1  $htmltext = '<div id="wrap"><h1>TEST</h1></div>';
2  echo htmlspecialchars($htmltext, ENT_QUOTE, "UTF-8");
```

<div id="wrap"><h1>TEST</h1></div>

これをブラウザで見ると、

<h1>TEST</h1>

となっている。

だから、フォームにて、JavaScript や <table> タグなどの余計な HTML タグが入力されたとしても、それを無力化できる。

2　PHP7+MySQL 入門ノートでの記述

『PHP7+MySQL 入門ノート』では、以下のような記述になっている。

リスト 1　discount.php (8-5)

```
1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4    <meta charset="utf-8">
5    <title>金額の計算</title>
6    <link href="../../css/style.css" rel="stylesheet">
7  </head>
8  <body>
9  <div>
10 <?php
11   require_once("../../lib/util.php");
12   // 文字エンコードの検証
13   if (!cken($_POST)){
14     $encoding = mb_internal_encoding();
15     $err = "Encoding Error! The expected encoding is " . $encoding ;
16     // エラーメッセージを出して、以下のコードをすべてキャンセルする
17     exit($err);
18   }
19   // HTMLエスケープ（XSS対策）
20   $_POST = es($_POST);
21 ?>
22   ( ...以下、略... )
```

この著者のやり方では、$_POST データが送られてきたら、まず、「文字エンコードの検証」をおこない (13 行目)、次に「HTML エスケープ」をおこなっている (20 行目)。

「文字エンコードの検証」は必要だと思うが、「HTML エスケープ」は、$_POST データを取得直後にするべきだろうか。

特に問題だと思われるのは、20 行目である。

$_POST = es($_POST)

$_POST の中味を書き変えている。

この es 関数がどのようなものかというと、

リスト 2　util.php

```
<?php
// XSS対策のためのHTMLエスケープ
function es($data, $charset='UTF-8'){
  // $dataが配列のとき
  if (is_array($data)){
    // 再帰呼び出し
    return array_map(__METHOD__, $data);
  } else {
    // HTMLエスケープを行う
    return htmlspecialchars($data, ENT_QUOTES, $charset);
  }
}
        ( ...以下、略... )
```

$_POST の中を再帰的に htmlspecialchars 関数を実行している。

たとえば、以下のような $_POST データが送られてきたとする。

```
 $_POST = [
  'name' => '<textarea>悪意</textarea>',
  'text' => '<script>alert("virus")</script>'
];
```

これを以下のコードで実行する。

```
<?php
require_once('util.php');

$_POST = [
  'name' => '<textarea>悪意</textarea>',
  'text' => '<script>alert("virus")</script>'
];

$_POST = es($_POST);
?>
<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title></title>
  </head>
  <body>
    <h1></h1>
    <h2>print_rで出力</h2>
    <pre><?php print_r($_POST); ?></pre>
    <h2>echoで出力</h2>
    <?php
    foreach($_POST as $key => $value) {
      echo  $key, ' ', $value, '<br>', PHP_EOL;
    }
    ?>
```

```
27      <script>
28       'use strict';
29
30      </script>
31    </body>
32  </html>
```

このようにブラウザに出力される。

print_r で出力

Array
(
  [name] => <textarea>悪意</textarea>
  [text] => <script>alert("virus")</script>
)

echo で出力
name <textarea>悪意</textarea>
text <script>alert("virus")</script>

しかし、実際は、以下のような文字列になっている。

```
<h2>print_r で出力</h2>
<pre>Array
(
  [name] => &lt;textarea&gt; 悪意&lt;/textarea&gt;
  [text] => &lt;script&gt;alert(&quot;virus&quot;)&lt;/script&gt;
)
</pre>
<h2>echo で出力</h2>
name &lt;textarea&gt; 悪意&lt;/textarea&gt;<br>
text &lt;script&gt;alert(&quot;virus&quot;)&lt;/script&gt;<br>
```

つまり、$_POST の中味がエスケープされた文字列に置き換っているのである。

ここでは、$_POST の中味をすぐに画面に出力しているからいいが、これを MySQL などに保存するとなると、大事になる。

3 このやり方の良くないところ

ここでの著者のやり方は、$_POST でデータが送られてきたら、とりあえず、htmlspecialchars 関数を使って $_POST を安全なものにしてしまおうというやり方である。

初心者の人にこのやり方を教えれば、この通りにすぐに htmlspecialchars 関数を使って同じようにやって

しまうだろう。

しかし、本来は、htmlspecialchars 関数は、画面に出力するタイミングで行うものでなければならない。この著者のやり方では、間違ったタイミングを教えてしまうことになる。

更に良くないのは、$_POST を書き変えてしまう点である。元のデータは大事にしなければならない。これは避けるべきだと思う。