

# 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	1
2	PHP7+MySQL 入門ノートでの記述	1
3	このやり方の良くないところ	4
4	この著者は実は、すぐに HTML エスケープすべきだとは思っていない	4
5	参考	7
6	じゃあ、どう書けばいいか？	7
6.1	util.php の es() 関数 . . . . .	7

## 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_QUOTES, "UTF-8");
```

```
&lt;div id=&quot;wrap&quot;&gt;&lt;h1&gt;TEST&lt;/h1&gt;&lt;/div&gt;
```

これをブラウザで見ると、

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

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、p266 で、「htmlspecialchars() を便利に使うためのユーザ定義関数 es()」として以下のような説明がある。

ユーザからのデータをブラウザに表示する前に htmlspecialchars() を通して HTML エスケープを行うことが必須となりますが、この処理を行うために array\_map() をうまく利用したユーザ定義関数を作っておくと便利です。

...(略)...

こうすることで、es() は引数が 1 個の値でも配列でも htmlspecialchars() で処理できる関数になります。

この説明のあとに、以下のコードが紹介されている。

リスト 1 util.php

```
1 <?php
2 // XSS対策のためのHTMLエスケープ
3 function es($data, $charset='UTF-8'){
4     // $dataが配列のとき
5     if (is_array($data)){
6         // 再帰呼び出し
7         return array_map(__METHOD__, $data);
8     } else {
9         // HTMLエスケープを行う
10        return htmlspecialchars($data, ENT_QUOTES, $charset);
11    }
12 }
```

その後に es() 関数をテストするコードが紹介されている。このコードを実行することで、htmlspecialchars 関数の働きを確認することができる。

で、この es() 関数を実際に使用したコードは、p272 の nameCheck.php である。

リスト 2 nameCheck.php

```

1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="UTF-8">
5   <title>フォーム入力チェック</title>
6   <link href="../../css/style.css" rel="stylesheet">
7 </head>
8 <body>
9   <div>
10
11   <?php
12     require_once("../../lib/util.php");
13     // 文字エンコードの検証
14     if (!cken($_POST)) {
15       $encoding = mb_internal_encoding();
16       $err = "Encoding Error! The expected encoding is " . $encoding ;
17       // エラーメッセージを出して、以下のコードをすべてキャンセルする
18       exit($err);
19     }
20     // HTMLエスケープ(XSS対策)
21     $_POST = es($_POST);
22   ?>
23
24   ... ( 以下、略 ) ...

```

ここでは、\$\_POST 連想配列の中の文字列を HTML エスケープした後、すぐに画面に出力している。

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、

\$\_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。

```

1 $_POST = [
2   'name' => '<textarea>悪意</textarea>',
3   'text' => '<script>alert("virus")</script>'
4 ];

```

これを以下のコードで実行する。

```

1 <?php
2 require_once('util.php');
3
4 $_POST = [
5   'name' => '<textarea>悪意</textarea>',
6   'text' => '<script>alert("virus")</script>'
7 ];

```

```

8
9 $_POST = es($_POST);
10 ?>
11 <!doctype html>
12 <html lang="ja">
13   <head>
14     <meta charset="utf-8"/>
15     <title></title>
16   </head>
17   <body>
18     <h1></h1>
19     <h2>print_rで出力</h2>
20     <pre><?php print_r($_POST); ?></pre>
21     <h2>echoで出力</h2>
22     <?php
23     foreach($_POST as $key => $value) {
24       echo $key, ' ', $value, '<br>', PHP_EOL;
25     }
26     ?>
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 <textarea> 悪意</textarea><br>
text <script>alert(&quot;virus&quot;)&lt;/script><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。

### 4 この著者は実は、すぐに HTML エスケープすべきだとは思っていない

この著者は、本当は、\$\_POST データを取得後すぐに HTML エスケープすべきだとは思っていないのは、以下のコードを見るとわかる。

p503 の "insert\_member.php" である。

リスト 3 insert\_member.php

```

1  <?php
2  require_once("../lib/util.php");
3  $gobackURL = "insertform.html";
4
5  // 文字エンコードの検証
6  if (!cken($_POST)) {
7      header("Location:($gobackURL)");
8      exit();
9  }
10
11 // 簡単なエラー処理
12 $errors = [];
13 if (!isset($_POST["name"]) || ($_POST["name"] === "")) {
14     $errors[] = "名前が空です。";
15 }
16 if (!isset($_POST["age"]) || (!ctype_digit($_POST["age"] === ""))) {
17     $errors[] = "年齢には数値を入れてください。";
18 }
19 if (!isset($_POST["sex"]) || !in_array($_POST["sex"], ["男", "女"])) {
20     $errors[] = "性別が男または女ではありません。";

```

```

21 }
22
23 // エラーがあったとき
24 if (count($errors) > 0) {
25     echo '<ol class="error">';
26     foreach ($errors as $value) {
27         echo "<li>", $value, "</li>";
28     }
29     echo "</ol>";
30     echo "<hr>";
31     echo "<a href=", $gobackURL, ">戻る</a>";
32     exit();
33 }
34
35 // データベースユーザ
36 $user = 'testuser';
37 $password = 'pw4testuser';
38 // 利用するデータベース
39 $dbName = 'testdb';
40 // MySQLサーバ
41 $host = 'localhost:8889';
42 // MySQLの DSN文字列
43 $dsn = "mysql:host={$host};dbname={$dbname};charset=utf8";
44 ?>
45
46 <!DOCTYPE html>
47 <html lang="ja">
48 <head>
49 <meta charset="utf-8">
50 <title>レコード追加</title>
51 <link href="../../../css/style.css" rel="stylesheet">
52 <!-- テーブル用のスタイルシート -->
53 <link href="../../../css/tablestyle.css" rel="stylesheet">
54 </head>
55 <body>
56 <div>
57     <?php
58     $name = $_POST["name"];
59     $age = $_POST["age"];
60     $sex = $_POST["sex"];
61     // MySQLデータベースに接続する
62     try {
63         $pdo = new PDO($dsn, $user, $password);
64         // プリペアドステートメントのエミュレーションを無効にする
65         $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
66         // 例外がスローされる設定にする
67         $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
68         // SQL文をつくる
69         $sql = "INSERT INTO member (name, age, sex) VALUES (:name, :age, :sex)";
70         // プリペアドステートメントを作る
71         $stm = $pdo->prepare($sql);
72         // プレースホルダに値をバインドする
73         $stm->bindValue(':name', $name, PDO::PARAM_STR);
74         $stm->bindValue(':age', $age, PDO::PARAM_INT);
75         $stm->bindValue(':sex', $sex, PDO::PARAM_STR);

```

```

76 // SQL文を実行する
77 if ($stm->execute()) {
78 // レコード追加後のレコードリストを取得する
79 $sql = "SELECT * FROM member";
80 // プリペアドステートメントを作る
81 $stm = $pdo->prepare($sql);
82 // SQL文を実行する
83 $stm->execute();
84 // 結果の取得(連想配列で受け取る)
85 $result = $stm->fetchAll(PDO::FETCH_ASSOC);
86 // テーブルのタイトル行
87 echo "<table>";
88 echo "<thead><tr>";
89 echo "<th>", "ID", "</th>";
90 echo "<th>", "名前", "</th>";
91 echo "<th>", "年齢", "</th>";
92 echo "<th>", "性別", "</th>";
93 echo "</tr></thead>";
94 // 値を取り出して行に表示する
95 echo "<tbody>";
96 foreach ($result as $row) {
97 // 1行ずつテーブルに入れる
98 echo "<tr>";
99 echo "<td>", es($row['id']), "</td>";
100 echo "<td>", es($row['name']), "</td>";
101 echo "<td>", es($row['age']), "</td>";
102 echo "<td>", es($row['sex']), "</td>";
103 echo "</tr>";
104 }
105 echo "</tbody>";
106 echo "</table>";
107 } else {
108 echo '<span class="error">追加エラーがありました。</span><br>';
109 }
110 } catch (Exception $e) {
111 echo "<span class="error">エラーがありました。</span><br>";
112 echo $e->getMessage();
113 }
114 ?>
115 <hr>
116 <p><a href="<?php echo $gobackURL ?>">戻る</a></p>
117 </div>
118 </body>
119 </html>

```

このコードでは、“insertform.html” から送られてきた \$\_POST['name'] データ、\$\_POST['age'] データ、\$\_POST['sex'] データについて、6 行目でエンコードチェックしているが、es() 関数は使っていない。

58~60 行目では、\$\_POST['name'] は \$name という変数に、\$\_POST['age'] は \$age という変数に、\$\_POST['sex'] は \$sex という変数に、格納している。その際にも、es() 関数は使っていない。

この \$name、\$age、\$sex は SQL 文に埋めこまれて、データベースに登録されている (73 行目 ~ 75 行目)。その際にも、es() 関数は使われていない。

es() 関数が使われているのは、99 行目 ~ 102 行目である。

この部分は、MySQL データベースから取り出したデータを画面に出力している。このデータはデータベースに登録されている全てのデータである。

それらのデータをデータベースに格納する際に、es() 関数を使って HTML 文字列や JavaScript を無効化させることはやっていない。あくまで、画面に出力する間際で es() 関数を実行している。

著者は、htmlspecialchars 関数の意味も使いどころもわかっているが、では、なぜ `$_POST = es($_POST)` なんてことを初心者におすすめしているのだろうか？

「初心者には難しいことを言ってもだめだから、とりあえず `$_POST` をエスケープさせとけ」ということだろうか？

htmlspecialchars 関数の使いどころは、PHP の初心者が一度は悩むところだと思う。その意味では、ここははっきりと、「画面出力の時点で htmlspecialchars 関数を使う」ということを明確にすべきではないかと思う。そして、その関数を使いやすくするために es() 関数などと、ユーザー定義関数をつくるのだということも。

## 5 参考

以下のサイトが参考になる。

脆弱性対策における htmlspecialchars() の使用箇所について

## 6 じゃあ、どう書けばいいか？

では、どう書けばいいか。

### 6.1 util.php の es() 関数

p267 に書かれている util.php の es() 関数であるが、`$_POST` を再帰的に htmlspecialchars() 関数を実行するなんてことは、必要ない。なぜなら、画面出力の時に htmlspecialchars() 関数にかけるのであるから、文字列に対してかけることになるからである。配列を配列のまま画面出力することはない。foreach() 関数などを使って、文字列を取り出して画面に出力するからである。

同じ名前の es() 関数だと紛らわしいので h() 関数とする。以下のようにする。

リスト 4 util.php

```
1 <?php
2 function h($data) {
3     return htmlspecialchars($data, ENT_QUOTES, "UTF-8");
4 }
```

Shist\_JIS や EUC、JIS を扱うことはないだろうから、"UTF-8" でいいのではないだろうか？

p272 の nameCheck.php は以下のようにする。

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4     <meta charset="UTF-8">
5     <title>フォーム入力チェック</title>
6     <link href="../../css/style.css" rel="stylesheet">
```



```

7 </head>
8 <body>
9 <div>
10
11 <?php
12     require_once("../lib/util.php");
13     // 文字エンコードの検証
14     if (!cklen($_POST)) {
15         $encoding = mb_internal_encoding();
16         $err = "Encoding Error! The expected encoding is " . $encoding ;
17         // エラーメッセージを出して、以下のコードをすべてキャンセルする
18         exit($err);
19     }
20     // <== <1>
21
22 <?php
23     // エラーフラグ
24     $isError = false;
25     // 名前を取り出す
26     if (isset($_POST['name'])) {
27         $name = trim($_POST['name']);
28         // <== <2>
29         if ($name === "") {
30             // 空白のときエラー
31             $isError = true;
32         }
33     } else {
34         // 未設定のときエラー
35         $isError = true;
36     }
37
38 <?php if ($isError): ?>
39     ... ( 略 ) ...
40 <?php else: ?>
41     <!-- エラーがなかったとき -->
42     <span>
43     こんにちは、<?php echo h($name); ?>さん。
44     </span>
45 <?php endif; ?>
46
47 </div>
48 </body>
49 </html>

```

<1> — h() 関数はここでは使わない。

<2> — \$\_POST['name'] を \$name に代入。

<3> — 画面に出力するときに h() 関数を使う。

この本の中では、いたる所で `$ _POST = es($_POST)` としているので、ほとんど書き直さねばならないということになる。