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、p266 で、「htmlspesialchars() 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     reuqire_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>
text <script>alert("virus")</script>

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か、以下のコードを見るとわかる。

p499 の”search.php” である。

リスト 3 search.php

```
1  <?php
2  require_once("../../lib/util.php");
3  $gobackURL = "searchform.html";
4
5  // 文字エンコードの検証
6  if (!cken($_POET)) {
7    header("Location:($gobackURL)");
8    exit();
9  }
10
11 // nameが未設定、空のときはエラー
12 if (empty($_POST)) {
13   header("Location:searchForm.html");
14   exit();
15 } else if (!isset($_POST["name"]) ||($_POST["name"] === "")) {
16   exit();
17 }
18
19 // データベースユーザ
20 $user = 'testuser';
```

```
21  $password = 'pw4testuser';
22  // 利用するデータベース
23  $dbName = 'testdb';
24  // MySQLサーバ
25  $host = 'localhost:8889';
26  // MySQLの DSN文字列
27  $dsn = "mysql:host={$host};dbname={$dbname"};charset=utf8";
28  ?>
29
30  <!DOCTYPE html>
31  <html lang="ja">
32  <head>
33  <meta charset="utf-8">
34  <title>名前検索</title>
35  <link href="../../css/style.css" rel="stylesheet">
36  <!-- テーブル用のスタイルシート -->
37  <link href="../../css/tablestyle.css" rel="stylesheet">
38  </head>
39  <body>
40  <div>
41    <?php
42    $name = $_POST["name"];
43    // MySQLデータベースに接続する
44    try {
45      $pdo = new PDO($dsn, $user, $password);
46      // 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
47      $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
48      // 例外がスローされる設定にする
49      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERROMDE_EXCEPTION);
50      // SQL文をつくる
51      $sql = "SELECT * FROM member WHERE name LIKE(:name)";
52      // プリペアドステートメントを作る
53      $stm = $pdo->prepare($sql);
54      // プレースホルダに値をバインドする
55      $stm->bindValue(':name', "%{$name}", PDO:PARAM_STR);
56      // SQL文を実行する
57      $stm->execute();
58      // 結果の取得(連想配列で受け取る)
59      $result = $stm->fetchAll(PDO::FETCH_ASSOC);
60      if (count($result) > 0) {
61        echo "名前に「($name)」が含まれているレコード";
62        // テーブルのタイトル行
63        echo "<table>";
64        echo "<thead><tr>";
65        echo "<th>", "ID", "</th>";
66        echo "<th>", "名前", "</th>";
67        echo "<th>", "年齢", "</th>";
68        echo "<th>", "性別", "</th>";
69        echo "</tr></thead>";
70        // 値を取り出して行に表示する
71        echo "<tbody>";
72        foreach ($result as $row) {
73          // 1行ずるテーブルに入れる
74          echo "<tr>";
75          echo "<td>", es($row['id']), "</td>";
```

```
76          echo "<td>", es($row['name']), "</td>";
77          echo "<td>", es($row['age']), "</td>";
78          echo "<td>", es($row['sex']), "</td>";
79          echo "</tr>";
80        }
81        echo "</tbody>";
82        echo "</table>";
83      } else {
84        echo "名前に「$name」は見つかりませんでした。";
85      }
86    } catch (Exception $e) {
87      echo "<span class="error">エラーがありました。</span><br>";
88      echo $e->getMessage();
89    }
90    ?>
91    <hr>
92    <p><a href="<?php echo $gobackURL ?>">戻る</a></p>
93  </div>
94  </body>
95  </html>
```

このコードでは、“searchform.html” から送られてきた $_POST['name'] 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。

42 行目では、$_POST['name'] を取り出して $name 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 は SQL 文に埋めこまれて、検索を実行している。

es() 関数が使われているのは、75 行目〜78 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。このデータは今 POST 送信されてきたデータではないが、過去において POST 送信されてきたデータであるだろう。

その送られてきた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すすめているのだろうか？

「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