htmlspecialchars 関数を使うタイミング v1.0

Seiichi Nukayama

2022 年 2 月 12 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のである。

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
23 <?php
24   // エラーメッセージを入れる配列
25   $errors = [];
26   //クーポンコード
27   if (isset($_POST['couponCode'])) {
28     $couponCode = $_POST['couponCode'];
29   } else {
30     // 未設定エラー
31     $couponCode = "";
```

```
32    }
33    //商品ID
34    if (isset($_POST['goodsID'])) {
35      $goodsID = $_POST['goodsID'];
36    } else {
37      // 未設定エラー
38      $goodsID = "";
39    }
40  ?>
41
42  <?php
43    // セールデータを読み込む
44    require_once("saleData.php");
45    // 割引率と単価
46    $discount = getCouponRate($couponCode);
47    $tanka = getPrice($goodsID);
48    // 割引率と単価に値があるかどうかチェックする
49    if (is_null($discount)||is_null($tanka)){
50      // 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
51      $err = '<div class="error">不正な操作がありました。</div>';
52      exit($err);
53    }
54  ?>
55
56  <?php
57    // 個数の入力値
58    if(isset($_POST['kosu'])) {
59      $kosu = $_POST['kosu'];
60      // 入力値のチェック
61      if (!ctype_digit($kosu)){
62        // 整数ではないときエラー
63        $errors[] = "個数は整数で入力してください。";
64      }
65    } else {
66      // 未設定エラー
67      $errors[] = "個数が未設定";
68    }
69  ?>
70
71  <?php
72  if (count($errors)>0){
73    //エラーがあったとき
74    echo '<ol class="error">';
75    foreach ($errors as $value) {
76      echo "<li>", $value , "</li>";
77    }
78    echo "</ol>";
79  } else {
80    // エラーがなかったとき（端数は切り捨て）
81    $price = $tanka * $kosu;
82    $discount_price = floor($price * $discount);
83    $off_price = $price - $discount_price;
84    $off_per = (1 - $discount)*100;
85    // ３桁位取り
86    $tanka_fmt = number_format($tanka);
```

```
87    $discount_price_fmt = number_format($discount_price);
88    $off_price_fmt = number_format($off_price);
89    // 表示する
90    echo "単価：{$tanka_fmt}円、", "個数：{$kosu}個", "<br>";
91    echo "金額：{$discount_price_fmt}円", "<br>";
92    echo "（割引：-{$off_price_fmt}円、{$off_per}% OFF)", "<br>";
93  }
94  ?>
95
96  <!-- 戻りボタンのフォーム -->
97    <form method="POST" action="discountForm.php">
98      <!-- 隠しフィールドに個数を設定してPOSTする -->
99      <input type="hidden" name="kosu" value="<?php echo $kosu; ?>">
100     <ul>
101       <li><input type="submit" value="戻る"></li>
102     </ul>
103   </form>
104
105 </div>
106 </body>
107 </html>
```

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)。

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しまっているのである。

この es 関数がどのようなものかというと、

リスト 2　util.php

```
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  }
13
14  // 配列の文字エンコードのチェックを行う
15  function cken(array $data){
16    $result = true;
17    foreach ($data as $key => $value) {
18      if (is_array($value)){
19        // 含まれている値が配列のとき文字列に連結する
20        $value = implode("", $value);
21      }
22      if (!mb_check_encoding($value)){
23        // 文字エンコードが一致しないとき
```

```
      $result = false;
      // foreachでの走査をブレイクする
      break;
    }
  }
  return $result;
}
// ?>
```

$_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;
    }
    ?>
    <script>
     'use strict';

    </script>
  </body>
</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　このやり方の危険なところ

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。

更に危険なのは、$_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。これは避けるべきである。