第４回　フォーム処理②

４－１．POST でのデータ送信

■POST データを送信

<form>開始タグ

method プロパティ	POST
action プロパティ	データの送信先を記載

■SampleCode（テキストボックスの内容を POST で送信）

「name="name"」　「name="tel"」　「name="address"」３件を POST で送信

```
<form action="sample01_4.php" method="POST">
  <label for="name" class="form-label">氏名</label>
  <input type="text" id="" class="form-control form-control-lg"
name="name">
  <label for="tel" class="form-label">電話番号</label>
  <input type="text" id="" class="form-control form-control-lg"
name="tel">
  <label for="address" class="form-label">住所</label>
  <input type="text" id="" class="form-control form-control-lg"
name="address">
  <div class="p-5 d-grid gap-2 d-md-flex justify-content-md-end">
    <button type="submit" class="btn btn-danger btn-lg">登録
    </button>
  </div>

</form>
```

■Sample（テキストボックスの内容を POST で送信・ブラウザ表示）　※入力画面

Sample 画面の[登録]ボタン押下時、POST 形式でデータが送信されます。

GET 形式と違って、遷移先（今回の Sample では sample01_4.php）の URL にパラメータは表示されません。（下記、赤枠部分）

4－2．POST データの受け取り

GET データ同様に、スーパーグローバル変数で受け取ることができます。

■POST データを取得できるスーパーグローバル変数

$_POST	HTTP POST 変数 HTTP POST メソッドから現在のスクリプトに渡された変数の連想配列です。

■SampleCode　1　（テキストボックスの内容を受け取り、変数に格納）

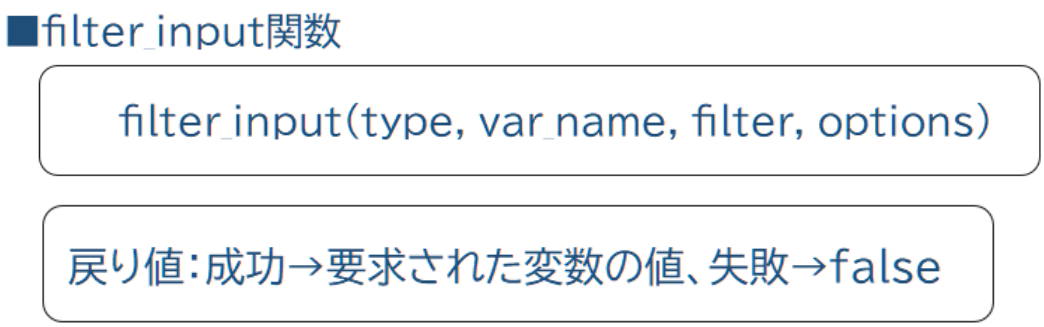
```php
<?php
  //パターン1

  //グローバル変数を各変数にそのまま取得

  $name = $_POST["name"];
  $tel = $_POST["tel"];
  $address = $_POST["address"];

?>
```

■SampleCode　2　（テキストボックスの内容を受け取り、連想配列に格納）

```php
<?php
  //パターン2

  //グローバル変数を連想配列にそのまま取得

  $result["name"] = $_POST["name"];
  $result["tel"] = $_POST["tel"];
  $result["address"] = $_POST["address"];

?>
```

スーパーグローバル変数では、送信されたデータをそのまま受け取るだけですので、続いて、値のフィルタリングを行ってみます。

■filter_input 関数



■引数

type	下記のいずれか INPUT_GET INPUT_POST INPUT_COOKIE INPUT_SERVER INPUT_ENV
var_name	取得する変数の名前
filter	適用するフィルタの ID。省略した場合は FILTER_DEFAULT。デフォルトでは、結果として、何もフィルタリングしません。 ■一部抜粋・概要 フィルタ ID / オプション / 説明 FILTER_VALIDATE_INT / default, min_range, max_range / 値が整数であるかどうか、オプションで指定した範囲内にあるかどうかを検証し、成功した場合は整数に変換します。 FILTER_VALIDATE_EMAIL / default / 値が妥当な e-mail アドレスであるかどうかを検証します。 **フィルタ ID や型の詳細は公式 HP 参照。** https://www.php.net/manual/ja/filter.filters.php
options	オプション指定が可能なフィルタの場合、フィールドフラグを指定します。 filter の値による。

■SampleCode　3　（filter_input 関数使用。テキストボックスの内容を受け取り、変数に格納）

※「name」と「address」は文字列が入ってくることが想定されるためフィルタなし。

※「tel」のみ int 型でフィルタ。（ハイフンは許容しない仕様と仮定。）

　電話番号なので最小値と最大値などのオプション指定なし。

```php
<?php
  //パターン3
  //filter_input 関数を用いて、変数に取得
  $name = filter_input(INPUT_POST, "name");
  $tel = filter_input(INPUT_POST, "tel", FILTER_VALIDATE_INT);
  $address = filter_input(INPUT_POST, "address");

?>
```

■SampleCode　4　（filter_input 関数使用。テキストボックスの内容を受け取り、連想配列に格納）

```php
<?php
  //パターン4
  //filter_input 関数を用いて、連想配列に取得
  $result["name"] = filter_input(INPUT_POST, "name");
  $result["tel"] = filter_input(INPUT_POST, "tel", FILTER_VALIDATE_INT);
  $result["address"] = filter_input(INPUT_POST, "address");

?>
```

■SampleCode　（HTML 部分）　※一部抜粋

※パターン１ or パターン３　それぞれ独立した変数に値を格納している場合

```
<p class="form-control form-control-lg"><?= $name ?></p>
<p class="form-control form-control-lg"><?= $tel ?></p>
<p class="form-control form-control-lg"><?= $address ?></p>
```

■SampleCode　（HTML 部分）　※一部抜粋

※パターン２ or パターン４　連想配列に値を格納している場合

```
<p class="form-control form-control-lg"><?= $result["name"] ?></p>
<p class="form-control form-control-lg"><?= $result["tel"] ?></p>
<p class="form-control form-control-lg"><?= $result["address"] ?></p>
```

■Sample（テキストボックスの内容を POST で受信・ブラウザ表示）　結果画面

※スーパーグローバル変数で値をそのまま取得した場合

※**filter_input 関数を使用していても、「tel」に数値のみ入力**されていた場合

■Sample（テキストボックスの内容を POST で受信・ブラウザ表示）　結果画面

※**filter_input 関数を使用し、「tel」に数値以外が入力**されていた場合

・入力「電話番号」

・結果

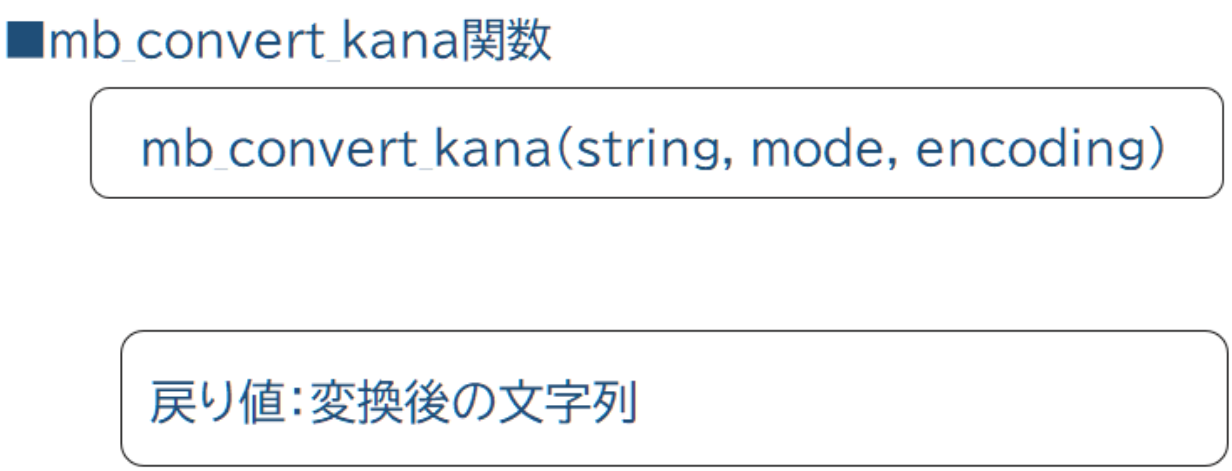
FILTER_VALIDATE_INT でフィルタリング失敗=false が返っている。

false は表示すると空文字になるので何も表示されていない。

続いて、文字列をプログラム上で扱いやすいように変換してみます。

■mb_convert_kana 関数

文字列の値を mb_convert_kana 関数を用いて、スペースは半角、カタカナは全角、濁点付きの文字は 1 文字に変換します。



■引数

<table>
<tr><td>string</td><td>変換される文字列</td></tr>
<tr><td>mode</td><td>変換オプション
<table>
<tr><td>r</td><td>英字「全角」→「半角」</td></tr>
<tr><td>R</td><td>英字「半角」→「全角」</td></tr>
<tr><td>n</td><td>数字「全角」→「半角」</td></tr>
<tr><td>N</td><td>数字「半角」→「全角」</td></tr>
<tr><td>a</td><td>英数字「全角」→「半角」</td></tr>
<tr><td>A</td><td>英数字「半角」→「全角」</td></tr>
<tr><td>s</td><td>スペース「全角」→「半角」</td></tr>
<tr><td>S</td><td>スペース「半角」→「全角」</td></tr>
<tr><td>k</td><td>カタカナ「全角」→「半角」</td></tr>
<tr><td>K</td><td>カタカナ「半角」→「全角」</td></tr>
<tr><td>h</td><td>ひらがな「全角」→「半角」</td></tr>
<tr><td>H</td><td>ひらがな「半角」→「全角」</td></tr>
<tr><td>c</td><td>「全角カタカナ」→「全角ひらがな」</td></tr>
<tr><td>C</td><td>「全角ひらがな」→「全角カタカナ」</td></tr>
<tr><td>V</td><td>濁点付きの文字を 1 文字に変換。"K", "H" と共に使用。</td></tr>
</table></td></tr>
<tr><td>encoding</td><td>文字列のエンコーディングを指定。省略可。省略した場合は、内部文字のエンコーディングを使用。</td></tr>
</table>

■SampleCode　※抜粋。

※スペースを半角、カナは全角に変換

```php
<?php
  //～(＊前述のコードは中略。)～
  //mb_convert_kana で「名前」「住所」を変換。
  // s→スペース「全角」→「半角」
  // K→カタカナ「半角」→「全角」
  // V→濁点付きの文字を 1 文字に変換。
  $result["name"] = mb_convert_kana($result["name"], "sKV", "UTF-8");
  $result["address"]
              = mb_convert_kana($result["address"], "sKV", "UTF-8");

?>
```

■Sample（mb_convert_kana 使用）

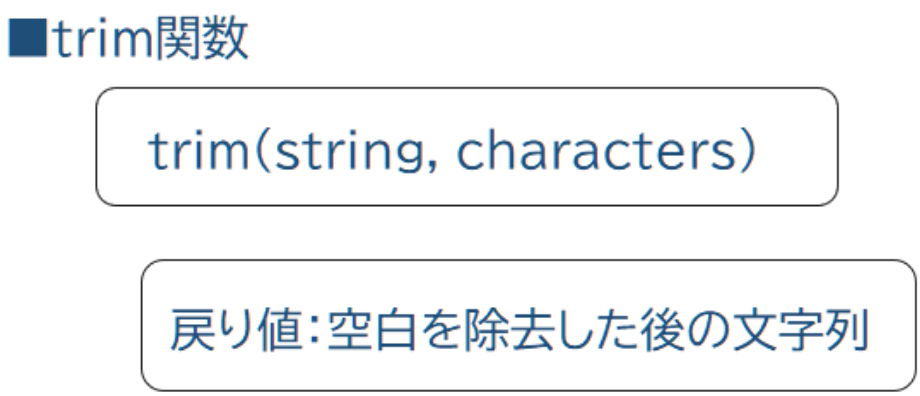
・入力

・結果

続いて、文字列の前後の空白の除去を行います。「　大阪市中崎町１－１０　」　のように前後に空白が入っていた場合、不要なので除去しておきます。「大阪市　中崎町　１－１０」のような文字と文字の間にある空白は除去されません。

■trim 関数

① 文字列に関しては、trim 関数を用いて、前後の空白を除去してください。



■引数

string	空白を取り除く文字列
characters	削除する文字列を指定することも可能。省略可。

■SampleCode　※抜粋。

```
<?php
  //～(＊前述のコードは中略。)～
  //mb_convert_kana の後に記述。

  //trim で「名前」「住所」の前後空白を除去。
  $result["name"] = trim($result["name"]);
  $result["address"] = trim($result["address"]);
?>
```

■Sample（trim 使用）

・入力

・結果