PHP基礎オンライン講座
2回目

2026/1/13

Kazuma SEKIGUCHI

前回のアジェンダ

- PHPとは何か
- PHPの動作環境
- PHPの記述方法
- 変数、文字列連結
- if文、ループ文
- 関数の利用
- ユーザー定義関数の作成

今回のアジェンダ

- POSTとGET
- ユーザーから送られてきたデータの受信
- データの妥当性チェック
- 正規表現
- セキュリティ対策について

ユーザーからデータをもらう

- 正確には、ユーザーがデータを送ってくる
  - 個人情報などは明示的に送信をして貰う必要がある
- 方法は2つ
  - GET送信、POST送信
- GET送信
  - URLの後ろに「?」を付けて特定のパラメータを送出する方法

https://www.google.co.jp/search?q=google&ie=utf-8&oe=utf-8

ユーザーからデータをもらう

- POST送信
  - フォームで送信するときに使われるタイプの送信方法
    - フォーム以外にもJSで生成してPOSTで送信してくることが可能
  - URLにパラメータは表示されない
    - GET送信と組み合わせることもできるため、組み合わせた場合は、GET送信のパラメータは表示される
  - ファイル（バイナリデータ）を送信可能

データの送信

パラメータ名 = 値

- データはパラメータ＝値として送信される
  - パラメータでデータの種類を識別する
- URLに含ませてデータを送信する
  - GET送信
  - aタグのhref属性で指定可能
- フォームを使ってデータを送信する
  - POST送信
  - フォームを使うか、JSでデータを組み立てて送信
- POST送信とGET送信を組み合わせることも可能

データの受け取り

- PHPの場合、ユーザから送信されてくるデータは全て特殊な変数に格納される（このときの変数名は大文字！！）
  - POST送信：$_POST
  - GET送信：$_GET
- 配列として格納されるため、使う場合は、$_GET[“パラメータ名”]、$_POST[“パラメータ名”]で取得する
  - postの場合はフォームのname属性の値がパラメータ名になる
- 必要なデータだけ取得するようにする
  - 全部を取得することも可能だが、セキュリティ的にまずくなる

サニタイズ

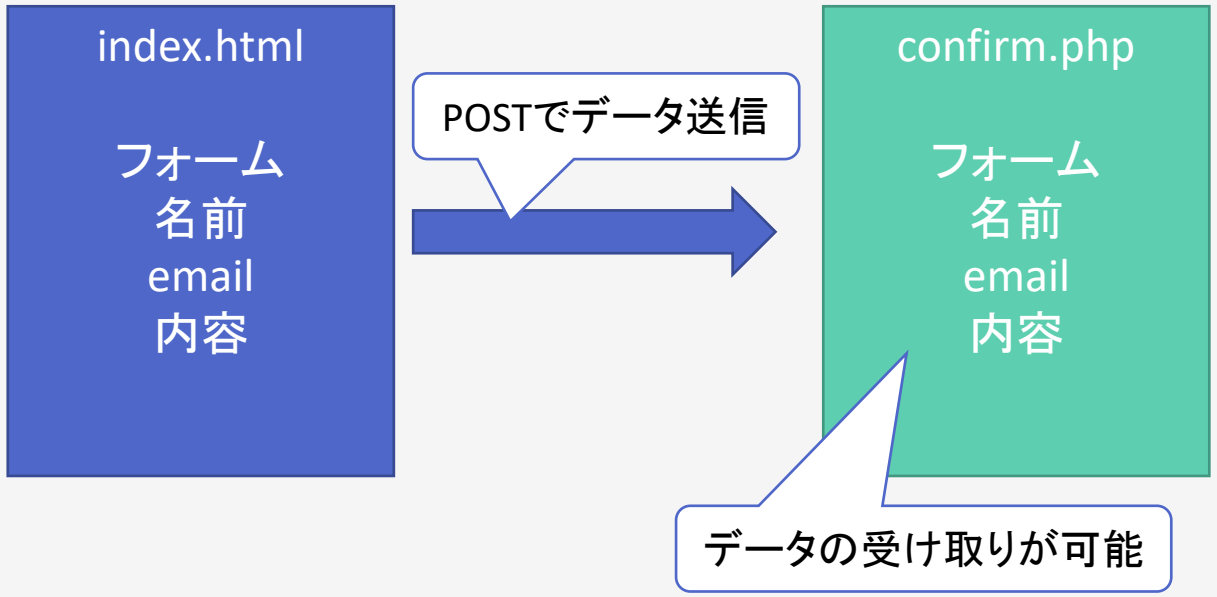
- サニタイズ＝無毒化
- ユーザが送ってくるデータが安全なデータとは限らない
  - パスワードを表示するような命令を送る
  - JavaScriptで永遠にウィンドウを開く命令を記述する

  などなど
- 送信されてきたデータが正しい形式（想定通り）かを検証する必要がある
  - そのまま利用しないこと
  - 基本的に送られてきたデータは信用しない（性悪説に基づく）

サニタイズ処理

- 数値かどうか判別する（強制的に数値にする）
  - 数値で判別することは多いため
  - intval()関数で数値に変換可能
- HTMLタグをタグとして利用できなくする
  - JavaScriptを動作させなくする
  - htmlspecialchars()関数で<と>を<や>に置き換える
- SQLインジェクション
  - DBに関係するが、DBのデータを不正に取得できないようにする（重要！）

PHPはデータがいる



- PHPはデータを与えられて処理が可能
  - フォームなどでデータを送る必要がある

データの受け取り

- 受け取ったデータは変数に格納する
  - 格納時にサニタイズ処理を行っておく
  - 変数とパラメータ名は一致している必要は無い
  - htmlspecialchars関数はHTMLタグを単なる文字列に変換する関数
    - XSS脆弱性を防ぐために必須

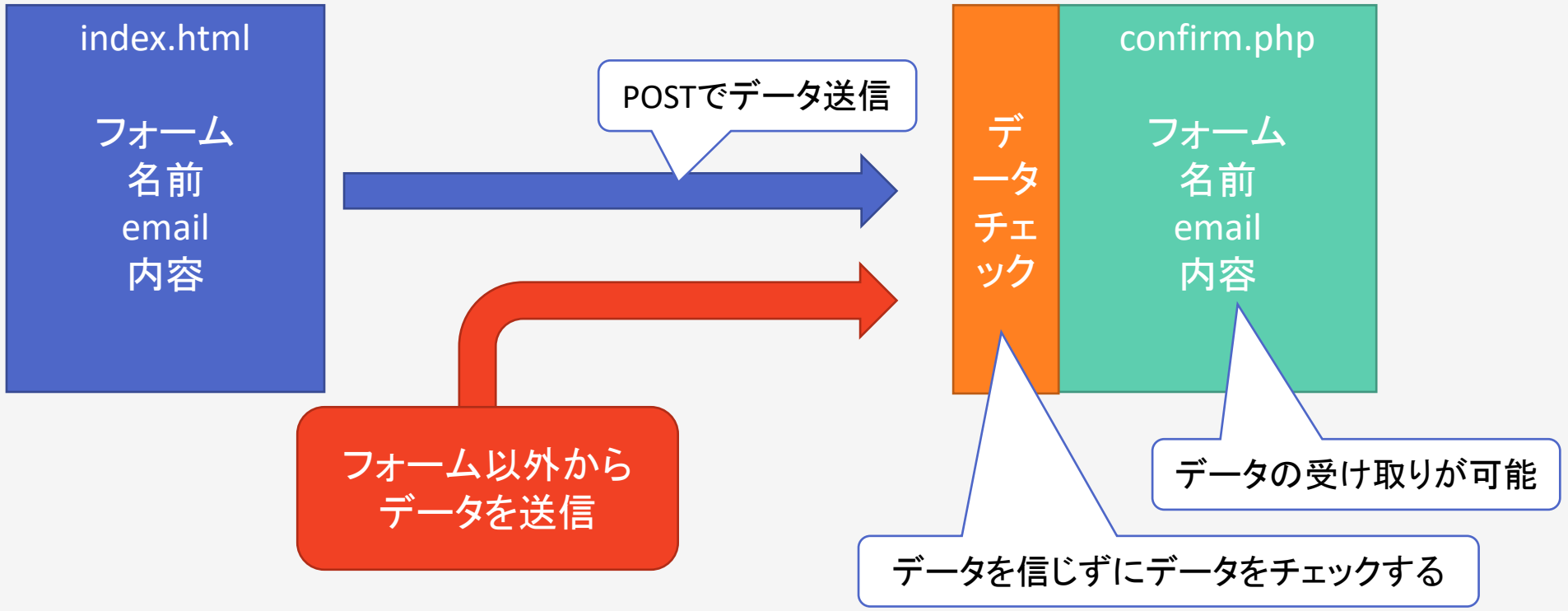
```
<?php
    $id = intval($_GET[“id”]);
    $age = intval($_POST[“age”]);
    $name = htmlspecialchars($_POST[“name”], ENT_QUOTES,”UTF-8“);
    $comment = htmlspecialchars($_POST[“comment”], ENT_QUOTES,”UTF-
    8“);
?>
```

isset関数

- 初期化されているかどうか判別してくれる関数
  - データが送信されてきているか、そうではないかを判別するときなどに利用する
  - $_POSTに何らかの値が入っている場合＝POST送信されてきている
- 同一ファイルに対してデータを送信しているかどうかで処理を分離することなどが可能

```
<?php
   if(isset($_POST)){
      //POST送信で何らかのデータが送られてきている
   }
?>
```

データはユーザーによって改変される



- データはフォーム以外から送信可能
  - データを正しいデータ形式として送信しなくても送信できる
  - 不正なデータ（サーバーを乗っ取るコードとか）を紛れ込ませることも可能

エラーのチェック

- フォームに入力された内容が正しいかどうかチェックしてダメならエラーとして返す
  - 名前：必須なら入っていればOK
  - ヨミガナ：カタカナ表記になっていることをチェック
  - 郵便番号：数字のみ（またはハイフンOK）
  - 住所：必須なら入っていればOK
- JSでチェックしていてもPHPでのチェックは必須
  - JSはチェックを切れる、チェックを避けることが可能なため
- ほとんどの場合は、正規表現でチェックする

正規表現（1）

- ユーザから入力してもらうデータは大体決まっている
  - メールアドレス、電話番号・・・などなど
- それぞれ決まり（パターン）がある
  - 電話番号：ハイフンと数字のみ
  - メールアドレス：英数字と一部の記号「@」必須
    @の直前に.が付かない
- このパターンが正しいかどうかチェックをすれば良い

正規表現（2）

- パターンをコンピュータに判別させるときの書式
- 英数字と記号を組み合わせてパターンを表現する
- ひらがなやカタカナも判別できるが、漢字は不可能
  （かなりトリッキーなことを使えば可能）

- テキストエディタなどで「検索」「置換」を行う際にも使える（便利！）

正規表現 PHPの場合

- 正規表現で入力されたデータが正しいか判断する場合
- preg_match関数を使用して引数に正規表現を記述
  - ereg関数は非推奨になっているので注意
- 正規表現は「/（スラッシュ）または@（アットマーク）」の間に記述する

```
    if(preg_match('/^[a-zA-Z0-9]+[a-zA-Z0-9¥._-]*[a-zA-Z0-9]+@[a-zA-Z0-9_-]+¥.[a-zA-Z0-9¥._-]+$/',$mail)){
            $str = '正しいメールアドレス';
    }
```

よく使う正規表現（1）

- ふりがな
  - ひらがなで書かれていることを検索

```
^[あ-ん]+$
```

- フリガナ
  - カタカナで書かれていることを検索
  - 半角カタカナを使わせないように全角カタカナだけを検索する

```
^[ァ-ンヴー]+$
```

小文字のア

よく使う正規表現（2）

- 電話番号、FAX番号、郵便番号
  - 数字とハイフンだけで書かれていることを検索
  - 電話番号は桁数が違う（携帯は11桁など）ことに注意

```
^[0-9-]+$
```

よく使う正規表現（3）

- メールアドレス
  - RFC 2822(http://tools.ietf.org/htmlrfc2822)
  - @があること。@の直前に「.」がないこと
  - 「.」が連続しないこと
  - 英数字と一部の記号で構成されていること（日本語ドメインを考慮すると異なる）

```
[A-Za-z0-9]{1}[A-Za-z0-9_.-]*[A-Za-z0-9]{1}@{1}[A-Za-z0-9]{1}[A-Za-z0-9_.-]{1,}¥.[A-Za-z0-9]{1,}$
```

よく使う正規表現（4）

- URL
  - RFC 3305及びその他で規定
    （http://tools.ietf.org/html/rfc3305）
  - 通常ユーザに入力させるのはHTTP
  - 英数字及び記号から構成される
    - 日本語ドメインもあるため、その場合はhttp、httpsが付いているなどで判断するのが現実的

```
^https?:¥/¥/[-_.!~*¥'()a-zA-Z0-9;¥/?:¥@&=+¥$,%#]+$
```

最近のチェック方法

- PHPでチェックするための関数が実装されている
  - 全てのパターンが実装されているわけでは無いため、足りない場合は正規表現を使って実装する
- メールアドレス

```
$result = filter_var($email, FILTER_VALIDATE_EMAIL, FILTER_FLAG_EMAIL_UNICODE);
                //$emailに格納したメールアドレスが正しければ、trueが返る
```

- URL
  - filter_varだけだとmailto:なども通ってしまうため、正規表現を組み合わせる

```
$result = filter_var($url, FILTER_VALIDATE_URL) && preg_match('@^https?+://@i', $url);
                //$urlに格納したURLが正しければ、trueが返る
```

チェック後

- エラーがあったときは、その旨を表示して修正して貰う
  - PHPで処理してしまうと入力した内容がクリアされてしまうため、フォームにユーザーが入力した内容を指定し、改めて入力して貰うのを防ぐ
  - inputタグのvalue値に指定すれば、入力状態になる
    - ファイルフィールドだけはvalue値に何も指定できない（セキュリティ上の問題）
  - ラジオボタンやチェックボックスならchecked属性を付与する

```
<input type="text" name="name" value="<?php print(htmlspecialchars($_POST["name"],ENT_QUOTES)); ?>">
```

```
<input type="radio" name="gender" value="女性" <?php if($gender === '女性'){ print('checked'); } ?>>
```

ありがとうございました。

また次回。