実践課題制作

2019/05/20

Kazuma Sekiguchi
class@cieds.jp

MAMPの設定が終わっていない場合（mac）

1. 「アプリケーション」⇒「MAMP」でMAMPを起動
2. MAMPメニューから「Preference」を選択
3. Portsタブで「Set Web & MySQL Ports to 80&3306」をクリックして、下部OKをクリック
4. MAMPフォルダー内のconfフォルダーを開く
5. PHP7.3.1フォルダーを開きphp.iniを以下のように書き換える

（Windowsも似たようなものですが、php.iniの書き換え箇所がちょっと違う）

php.iniの設定(Macの場合)

472行目付近

display_errors = Off → display_errors = On

665行目付近

post_max_size = 8M → post_max_size = 32M

データ量として受け入れる最大サイズ。ファイルサイズでは無く、パケットの総合計サイズになる

910行目付近

date.timezone = “Europe/Berlin” → date.timezone = Asia/Tokyo

時間を出すときに利用される。日本であれば、GMT+9なので、タイムゾーンを変えないと変な時間が使われる

php.iniの設定（Macの場合）

1597行目付近以下

文字コードの設定。変更しないと文字化けする

;mbstring.language = Japanese	mbstring.language = Japanese
;mbstring.internal_encoding = EUC-JP	mbstring.internal_encoding = UTF-8
;mbstring.http_input = auto	mbstring.http_input = pass
;mbstring.http_output = SJIS	mbstring.http_output = pass
;mbstring.encoding_translation = Off	mbstring.encoding_translation = Off
;mbstring.detect_order = auto	mbstring.detect_order = UTF-8,SJIS,EUC-JP,JIS,ASCII

php.iniの設定（macの場合）

- 変更をしたら上書きして保存する
- ファイルを保存したらMAMPのパネルから「Stop Servers」を押してサーバーを停止
- 「Start Servers」を押して再度サーバーを起動
  - 問題なく起動すればOK
  - 起動しない場合、設定をどこか間違えている可能性があるため、再度確認する（セミコロンを外す、残しておくとか、スペルなど）

Apacheの起動(mac)

セキュリティに引っかかってしまうので

- 「アプリケーション」⇒「ユーティリティ」⇒「ターミナル」を起動
- 「アプリケーション」⇒「MAMP」⇒「bin」⇒「apache2」⇒「bin」
- 「ターミナル」でshと入力し、スペース
- 「bin」内にあるapachectlを「ターミナル」のshの後（スペースの後）にドラッグ＆ドロップ
- スペースを入れて、startと入力
  $ sh /Applications/MAMP/Library/bin/apachectl start
  となるはず
- これでReturnキーを押して何も出なければApacheが起動する
- 自分のマシンであれば、MAMPの画面から起動させればOK

ユーザからデータをもらう

- 正確には、ユーザがデータを送ってくる
  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
    - 複数を送出する場合、2つ目以降のパラメータは「&」で繋ぐ
  - 手軽であるが、文字数制限がある（最近のブラウザには無い）

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

ユーザからデータをもらう

- POST送信
  - 通常フォームで送信するときに使われるタイプの送信方法
  - 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

フォームの形式

- ファイルをフォームからアップロードしてもらう場合、enctypeを変更する必要がある
  - 変更しないとファイルがアップロードされないので注意

```
<form action="confirm.php" method="post" enctype="multipart/form-data" name="form1" id="form1">
```

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！）

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
- 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

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

```
<?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“);
?>
```

メールの送信

- HTMLで入力して貰うためのフォームは作成可能
  - 但し、メールでフォームに入力された内容を送る機能は無い
  - 1度サーバにデータを送り、サーバでデータをメールに変換して送り出す
    - サーバで処理するための仕組みが必要
    - サーバで処理＝サーバサイドプログラム

- PHP、Ruby、Perl、Pythonなどが良く使われる
  - PHPが比較的採用例も多く、簡単なものなら記述も楽

フォーム

- HTMLで作成されるユーザが入力できる唯一の画面
- 主要なタグ
  - <form>：フォームの開始を示す
  - <input>：ほとんどのフォーム部品
  - <textarea>：長いテキストを入力して貰うときの部品
  - <label>：部品の説明（ex：氏名など）

- ほとんどはinputタグのtype属性で変化させる
- name属性に固有の値を設定しておく
- 送信先はformのaction属性で指定する

最小限のフォーム

```
<form action=“mail.php” name=“form1” method=“post”>
<input type=“text” name=“name”>
<input type=“submit”>
</form>
```

- formのmethod属性：postかgetを選択。通常はpost
- action属性：データの送信先を指定
- input type=“text”：テキストフィールドの作成
- input type=“submit”：送信ボタンの作成
  - クリックすることで、テキストフィールドに入力されたデータが送信される

フォームの注意

- 必須などのチェックを行う機能は無い
  - HTML5から必須チェックを行うことが可能（ブラウザ依存）
  - JSでチェックを行う
- 正しいデータが入っているかは調べられない
  - 形式に沿っているかどうかはJSで判別可能
    - 電話番号、メールアドレスなど
- 送信ボタンを押すと、データの送信先に遷移する
  - 確認画面などはデータの送信先のプログラム上で作成する必要あり
  - JSを利用して、データだけを送ることも可能（遷移無し）

フォームの部品

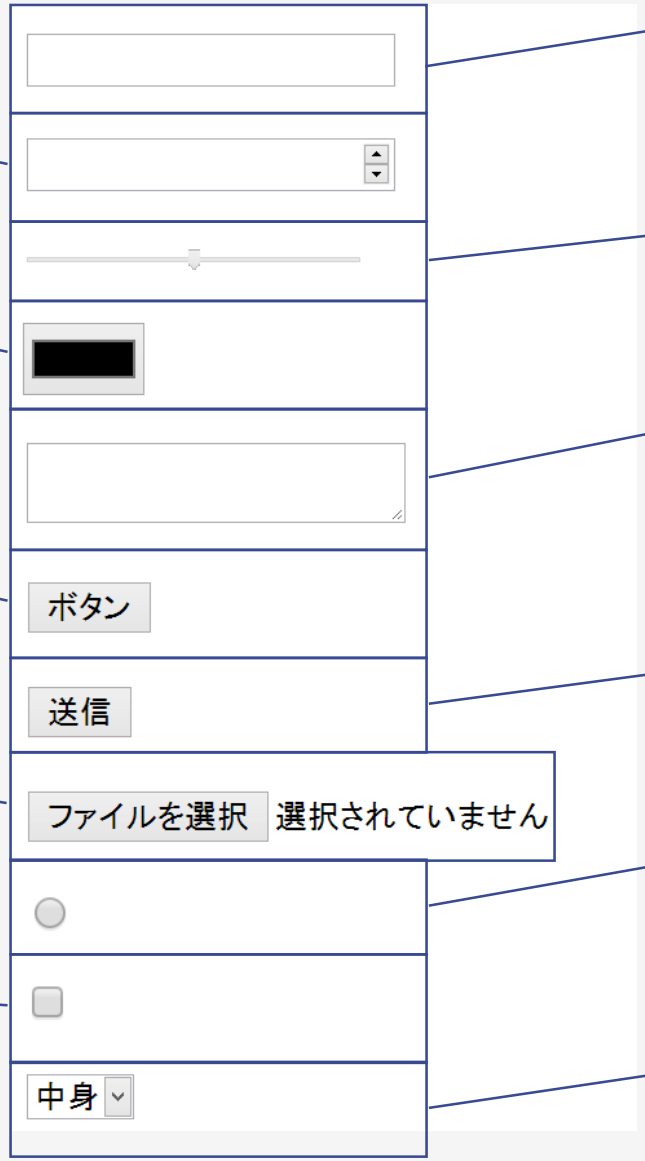
数値フィールド(HTML5)
<input type="number">

色選択(HTML5)
<input type="color">

ボタン
<input type="button">

ファイル
<input type="file">
保存されているファイルを
アップロードする場合に使用

チェックボックス
<input type="checkbox">



テキストフィールド
<input type="text">

範囲指定(HTML5)
<input type="range">

テキストエリア
<textarea>

送信ボタン
<input type="submit">

ラジオボタン
<input type="radio">
排他処理

セレクトボタン
<select>
中身は<option>で指定

フォームとCSS

- フォームの部品は全てCSSを適用可能
  - 色や枠線などもCSSで規定されているため、指定することで上書き可能
  - フォーム部品自体はインライン要素

- CSSで記述する場合は、属性値を含めた指定をすると指定しやすい

```
input[type=“text”]{
    background-color:#FFF;
}
```

テキストフィールドのみに適用

```
input[type=“submit”]{
    background-color:#FFF;
}
```

送信ボタンのみに適用

メール送信

- PHP内部にあるmail関数を利用
  - 添付ファイルなどをしたい場合は、PHPMailerなどの外部ライブラリを利用するのが一般的
    - 外部ライブラリを使うことで、同じ処理をまた書かなくて良い
    - 多量のテストがされているため、バグが発生しない

メール送信

- 単なるテキストメールであれば、以下のコードで送信可能
  - マルチバイト関数が利用できることが条件
  - マルチバイト関数が利用できないと文字化けが生じる
- メール内での改行は「”￥n”」で記述する
  - コード内で改行しても意味が無い。また「’」では改行されない

```
mb_language("japanese");
mb_internal_encoding("utf-8");
$to = $email;
$subject = "フォームからのメール";//メールのタイトル
$body = 'フォームからのメール'."¥n".'名前：'.$name."¥n".'性別：'.$gender."¥n".'メールアドレス
'.$email."¥n".'内容'.$contents;
$from = "from@example.com";//メール差出人アドレス
mb_send_mail($to , $subject , $body , "From:".$from);//メールを送信
```

あまったら

フォームのチェック

- ユーザに入力して貰う場面は意外に多い
  - ECサイトでの買い物

- 正しいデータが必要
  - 買って貰ったのに送れない・・・

- 入力を間違えると処理するプログラムのせいで、入力したデータが全て消えることがある（昔は良くあった）
  - ユーザの負担増

フォームのチェック（2）

- JavaScriptで予め入力データをチェック
  - 不足しているデータや間違えを指摘
  - ほぼリアルタイムで判別出来るので、ユーザの負担が減る
  - 入力したデータが全て消えることがない

- JavaScriptでチェックしてもサーバサイドで再度のチェックは必要
  - JavaScriptはユーザが任意にOffにできる
  - 受け取ったデータは必ずPHP側でもチェックする
  - HTML5のフォームチェックも回避が可能

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(3) 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）」の間に記述

```
    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-ヾ]+$
```

小文字のア

踊り字（おどりじで変換）

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
    - DocomoとAuでは許可している
  - 英数字と一部の記号で構成されていること（今後変わるかも）

```
^[a-zA-Z0-9._-]+[a-zA-Z0-9_]*@[a-zA-Z0-9._-]+¥. [a-zA-Z0-9._-]+$
```

ちなみに

- 厳密にメールアドレスを正規表現で表すと

```
/^(?!(?:(?:¥x22?¥x5C[¥x00-¥x7E]¥x22?)|(?:¥x22?[^¥x5C¥x22]¥x22?)){255,})(?!(?:(?:¥x22?¥x5C[¥x00-¥x7E]¥x22?)|(?:¥x22?[^¥x5C¥x22]¥x22?)){65,}@)(?:(?:[¥x21¥x23-¥x27¥x2A¥x2B¥x2D¥x2F-¥x39¥x3D¥x3F¥x5E-¥x7E]+)|(?:¥x22(?:[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x21¥x23-¥x5B¥x5D-¥x7F]|(?:¥x5C[¥x00-¥x7F]))*¥x22))(?:¥.(?:(?:[¥x21¥x23-¥x27¥x2A¥x2B¥x2D¥x2F-¥x39¥x3D¥x3F¥x5E-¥x7E]+)|(?:¥x22(?:[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x21¥x23-¥x5B¥x5D-¥x7F]|(?:¥x5C[¥x00-¥x7F]))*¥x22)))*@(?:(?:(?!.*[^.]{64,})(?:(?:(?:xn--)?[a-z0-9]+(?:-[a-z0-9]+)*¥.){1,126}){1,}(?:(?:[a-z][a-z0-9]*)|(?:(?:xn--)[a-z0-9]+))(?:-[a-z0-9]+)*)|(?:¥[(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){7})|(?:(?!(?:.*[a-f0-9][:¥]]){7,})(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,5})?::(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,5})?)))|(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){5}:)|(?:(?!(?:.*[a-f0-9]:){5,})(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,3})?::(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,3}:)?)))?(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-9]{2})|(?:[1-9]?[0-9]))(?:¥.(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-9]{2})|(?:[1-9]?[0-9]))){3}))¥]))$/iD
```

だそうです

（http://emailregex.com/より）

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（今後日本語も）

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

リダイレクト

- データを受け取った後で処理を行い、処理が完了したら、別のページに飛ばすことがある

- PHPのheader関数を利用することで任意のページに飛ばすことが可能
  - ただし、何か表示するような命令を記述した後でheader関数を呼び出してもエラーになるので注意

header関数

- 任意のHTTPヘッダを送出する関数
  - HTTPヘッダにブラウザーを別のページへと遷移させるためのものがある
  - header関数を利用することで、任意のページへと飛ばすことが可能
  - 遷移させたいページは絶対パス（URL）で記述するのが正しいが、相対パスでも遷移はする

```
<?php
header('Location:絶対パス');
?>
```