実践課題制作

2019/05/27

Kazuma Sekiguchi
class@cieds.jp

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');
?>
```

ファイルのアップロード

- Webの場合、ファイルをアップロードしてもらうのは画像ファイルなどに止めておくべき
  - GIF，PNG，BMP，JPEGなどのみ有効にする
  - WindowsやmacOSで使われているファイル形式以外も有効なので、注意
  - たとえば、PHP、Perlの実行形式など

ファイルのアップロード

- 指定したサーバ上のフォルダにアップロード
  - PHP上では、一時フォルダから指定のフォルダにコピーという形を取る
  - アップロードされたファイルには別名を付ける
    - 万が一同じファイル名のファイルがあった際のことを想定
    - 日時秒を付与したり、ランダム関数を使って文字列を付与

ファイル形式のチェック

- アップロードされたファイルの形式チェックは必須
  - 実行ファイル形式をアップされていないかチェックする
  - 拡張子によるチェックが一般的だが、確実では無い
  - 画像ファイルの場合、PHPの関数による画像形式チェックを併用した方が良い

ファイル形式のチェック

- 画像の場合画像の形式をチェック
- exif_imagetype関数が便利

```
<?php
if(file_exists($file_pass) && $type = exif_imagetype($file_pass)){
    switch($type){
        case IMAGETYPE_GIF: //gifの場合
        echo "IMAGETYPE_GIF";
        break;
        case IMAGETYPE_JPEG:  //jpgの場合
        echo "IMAGETYPE_JPEG";
        break;
        case IMAGETYPE_PNG:  //pngの場合
        echo "IMAGETYPE_PNG";
        break;
        default:   //どれにも該当しない場合
        echo "gif、jpg、png以外の画像です";
    }
}else{
    echo "画像ファイルではありません（もしくはファイルが存在しません）";
}
?>
```

画像ファイルの保存

- ファイルがアップロードされたきた場合
  1. ファイルの形式をチェック
     - 拡張子で判断（gif ， jpeg ， pngなど）
     - それ以外は全て撥ねる。実行可能形式をアップロードされたらサーバが乗っ取られる
     - 画像の場合は、exif_imagetype()関数を利用して、画像形式を取得する方がよりベター（拡張子のチェックだけでは不安）
  2. サーバに一時的に保存されているので、一時的な名前を取得し、新しいファイル名を付ける（万が一同じファイル名があった場合に上書き処理になってしまうため）

画像ファイルの保存

- ファイルがアップロードされたきた場合
  3. ファイルを新しい名前に変更して一時保存フォルダから移動する
     - move_uploaded_file（）関数を利用
     - LinuxなどのUnix系サーバでは、ファイルパーミッションを書き込み可能にすること
  4. ファイル名は新しく付けたので、それを利用して<img>タグなどで呼び出せば、確認画面を作成可能
     - 画像をリサイズする場合は、GD関数を利用する
     - リサイズする場合は縦横比に注意

いろいろな画像形式に対応する場合

- PHPの場合、画像形式に合わせて使用するimageCreateFromXXXを選ぶ必要がある
  - 形式を取得して使用するimageCreateFromXXXを返すユーザー定義関数を作成すると使い回しができて楽

```
function imageCreateFromAny($filepath) {
    $type = exif_imagetype($filepath);
    $allowedTypes = array(
        1,  // [] gif
        2,  // [] jpg
        3,  // [] png
        6   // [] bmp
    );
    if (!in_array($type, $allowedTypes)) {
        return false;
    }
    switch ($type) {
        case 1 :
            $im = imageCreateFromGif($filepath);
        break;
        case 2 :
            $im = imageCreateFromJpeg($filepath);
        break;
        case 3 :
            $im = imageCreateFromPng($filepath);
        break;
        case 6 :
            $im = imageCreateFromBmp($filepath);
        break;
    }
    return $im;
}
```

縦横比を維持して指定するサイズに縮小

- $imageに画像ファイル名を指定、$maxwidthと$maxheightにそれぞれ算出したいサイズを指定すれば、縦横比を維持したまま縮小したサイズを算出して返す

```
function image_ration($image , $maxwidth , $maxheight){
if(file_exists($image)){
$img1_arr = getimagesize($image);
//var_dump($img1_arr);
if($img1_arr[0] > $maxwidth || $img1_arr[1] > $maxheight){
if($img1_arr[0] > $maxwidth){
$img1_width_ratio = $maxwidth / $img1_arr[0];
}
else{
$img1_width_ratio = 1;
}
if($img1_arr[1] > $maxheight){
$img1_height_ratio = $maxheight / $img1_arr[1];
}
else{
$img1_height_ratio = 1;
}
$img1_ratio = min($img1_width_ratio , $img1_height_ratio);
$img1_width = intval($img1_ratio * $img1_arr[0]);
$img1_height = intval($img1_ratio * $img1_arr[1]);
}
else{
$img1_width = $img1_arr[0];
$img1_height = $img1_arr[1];
}
return array($img1_width,$img1_height);
}else{
return array(0,0);
}
}
```

テキストファイルの保存

- テキスト情報を保存する場合
  1. サニタイズ処理を行っておく
  2. 指定したファイルを開く
     - fopen()関数で開く。第一引数で開くファイル名、第二引数で開き方を指定可能
     - “a”を指定するとファイルが無ければ新規に作成し、追記モードで開かれる（一番楽）
  3. 書き込む内容を生成する
     - テキストの場合は、CSV書式がベターと思われる
     - 1つの書き込みに付き、1行を利用する

テキストファイルの保存

- テキスト情報を保存する場合
  4. 実際に書き込むには、fwrite()関数を利用
     - 第一引数にfopen()関数の返値を指定、第二引数に書き込む内容を指定する

- 読み出す場合
  - fopen()関数で、”r”を指定し、読み出しモードにする
  - ループを利用して、!feof($fp)に到達するまでループを利用して中身を取り出す

テキストファイルの保存

- 読み出す場合
  1. fopen()関数で、”r”を指定し、読み出しモードにする
  2. ループを利用して、!feof($fp)に到達するまでループを利用して中身を取り出す
  3. fgets()関数でファイルから1行取り出すことが可能。これを終わりまで繰り返せば、全て取得することが可能