PHP基礎オンライン講座
3回目

2026/1/20

Kazuma SEKIGUCHI

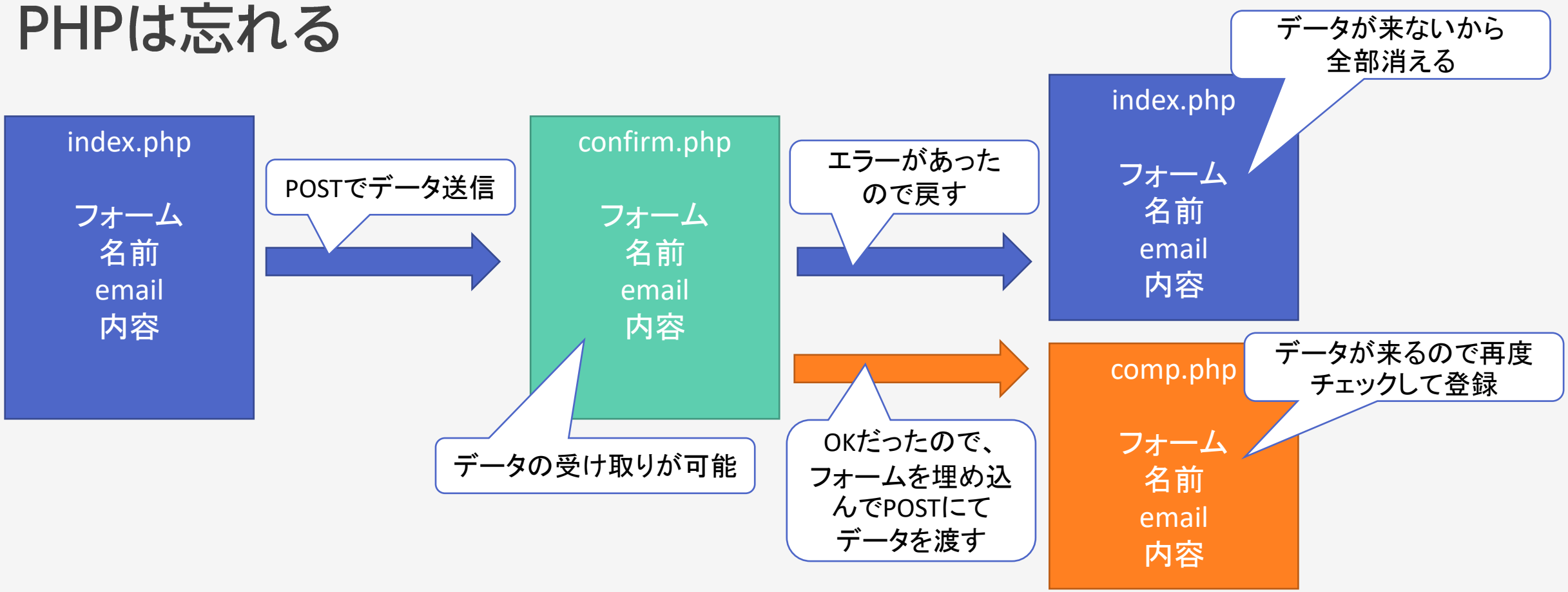
前回のアジェンダ

- 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

今回のアジェンダ

- セッションとは何か
- フォームでのエラーへの対応
- 画像のアップロードと表示
- メールの送信
- HTTPヘッダの出力
- ファイルへの保存、読み出し

PHPは忘れる



- PHPは別のファイルに移動するときにデータを渡さないとデータを忘れる
  - confirm.php（確認画面）からcomp.php（完了画面）へもデータを渡す必要がある
  - input type=“hidden”で渡すのが一般的だが、再びcomp.phpでデータの確認が必要

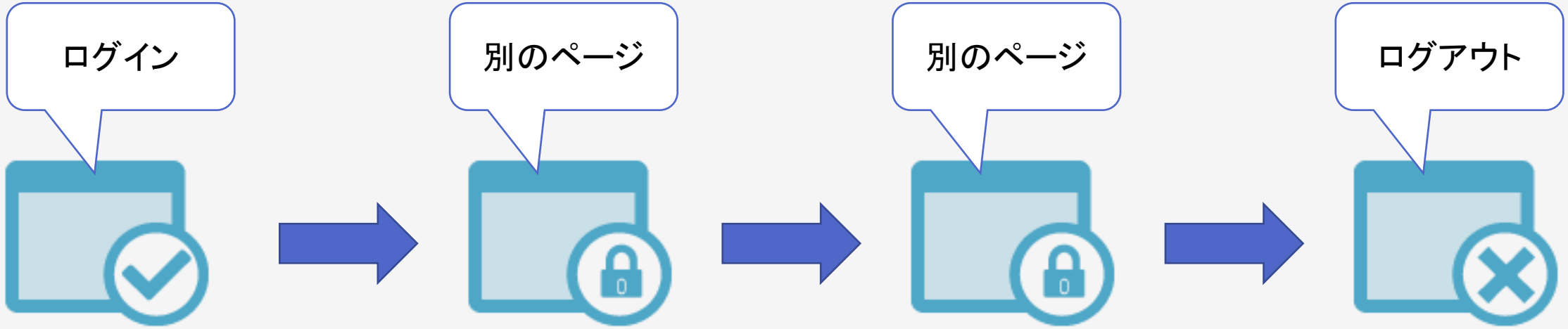
セッション

- 通常のプログラムでは、状態遷移を把握することが必要
  - 今ユーザはどこを見ているのか？
  - どんな状態か？

- 状態を把握できないと、プログラムを制御できない
  - プログラムを終了したのにいつまでも実行しているといったことが起こる
  - 通常状態遷移はアプリケーション側で実装する（イベントはOSから送られる）
  - 管理画面など認証が必要な画面では必須な要素

セッション

- ユーザがログインして、ページを利用した後でログアウトするまでの一連の流れ
  - ユーザ別に違う機能やページを表示する場合、認証が必要
  - 認証してその情報に基づいてページを表示する

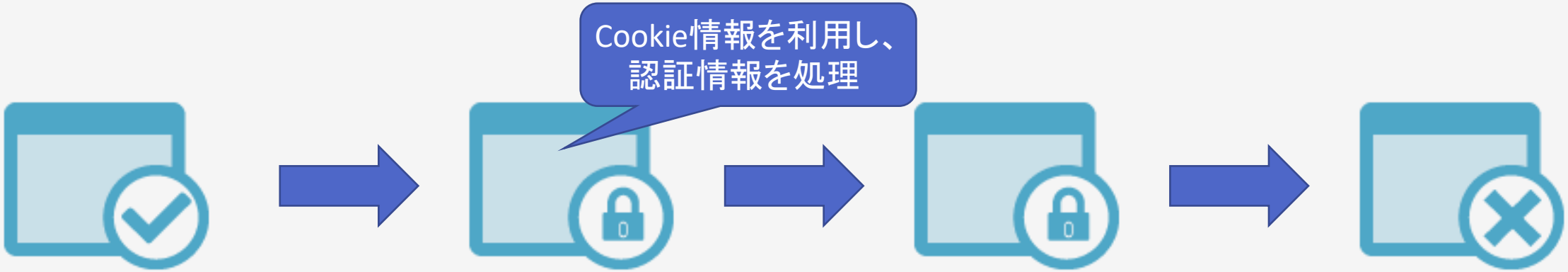


セッション

- 認証したら、きちんと認証したという状態を保持しておく必要がある
  - 保持していないと毎回認証していない、ということで認証させる必要が生じる
  - HTTPはステートレス＝認証状態なんて保持してくれない
  - ページが変わった瞬間に認証した、という事実も失われる

セッション管理

- 認証したら認証した事実を保持する仕組み＝セッション管理
  - 通常はCookieを用いて、クライアント側にあるデータを保持させる
    - Cookieは小さいデータを保持することが可能
    - GET送信する際にCookieのデータを同時に送信する
  - サーバはCookieのデータを判断し、状態を判断することが可能
    - 例えば、cookieから所定のデータが来ていればログインしている状態、と判断する（プログラムによっても異なる）



Cookie

- Cookieは元の送信元サーバにしかデータを送信しない
  - URLで管理
    - インラインフレームなどから取得することは不可能
  - セキュリティ保持のため
- JSから作成することも可能
  - JSから読み取ることも可能（同じURLから送信されたJSに限る）
- 期限なども設定可能
  - 期限が来たら、自動的に削除される
  - プログラムで削除することも可能
- 広告会社などは広告にCookieデータを含ませて送信し、どんなページを見ているか取得している
  - ブラウザー設定でブロック可能

セッションの動き

1．ユーザがIDとパスワードを送信

5．クライアントは受け取ったデータを表示し、セッションIDをCookieに保存する

6．次のページの要求時にセッションIDを同時に送信

2．IDとパスワードを確認

3．正しければ、ログインできたパラメータを固有のセッションIDに紐づけて、サーバに保存

4．クライアントにデータとセッションIDを返す

7．セッションIDを受け取ったら、保存されているパラメータをサーバから取り出す

8．パラメータに従ってプログラムが処理する

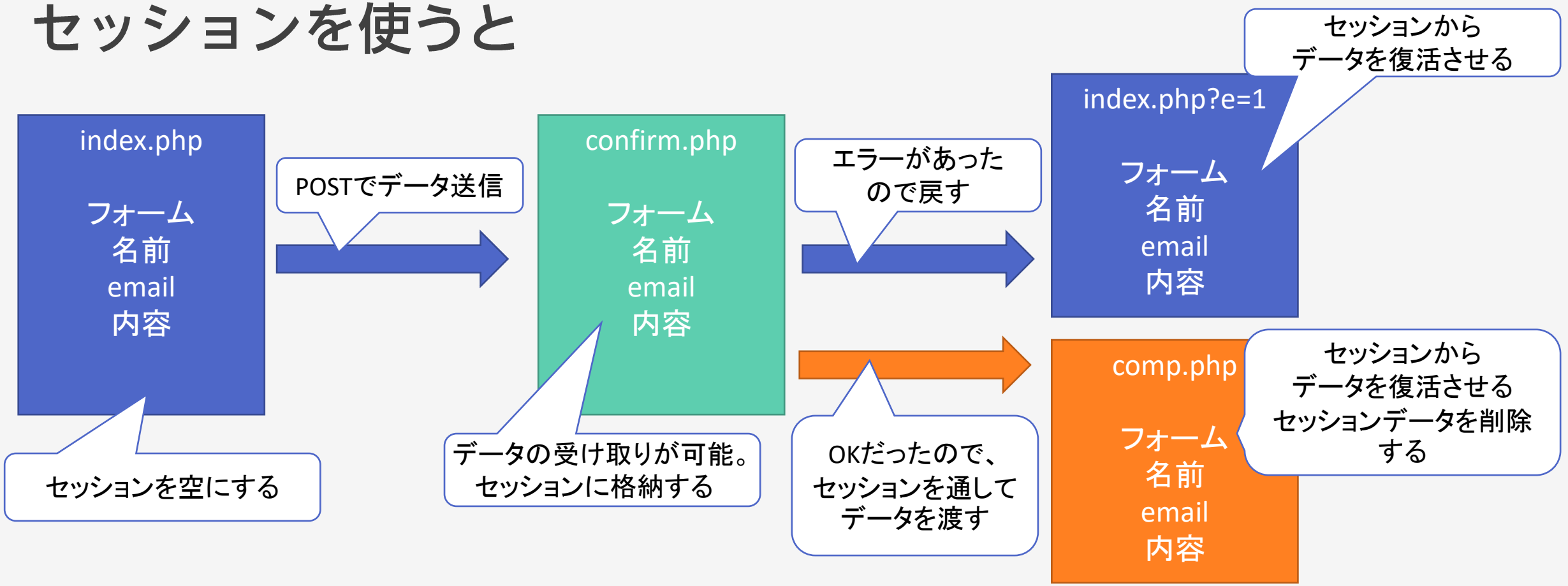
PHPからセッションを使う

- セッション管理をしたいすべてのページにsession_start();関数を入れておく
- ログアウトなどの際には、session_destroy();関数を使用して、セッションデータを消去する
  - データが無い＝ログインしていない
- 実際のところ、ある程度のデータを半永久的に保存しておくために使われる
  - フォームでのデータを格納しておけば、別ページでも取り出せる

```
<?php
session_start(); //セッション管理を開始する場合に呼び出す関数。必ず最初に書く。これよりも前に何か処理を書いたりすると機能しない

$_SESSION['パラメータ名'] = 値;//この形式で保存が可能
$変数名 = $_SESSION['パラメータ名'];//この形式で取り出しが可能
```

セッションを使うと



- confirm.phpでセッションにデータを格納
  - エラーで戻したときはセッションからデータを復活させることで入力の手間を減らす（URLでセッションから復活させるかどうかを識別させることは必要）

ファイルアップロード

- 画像ファイルなどをフォームを通してアップロードして貰うことが可能

- セキュリティ上のリスクが高いため、使用には注意が必要
  - 実行ファイル（PHPファイルやPerlのファイルなど多種）をアップロードされる可能性があることに注意
- 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やzip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

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は元のファイル名（ユーザーが付けていた名前）を取得可能
  - アップロードされたファイルには別名を付ける
    - 多数のユーザーが使った場合、同じ名前のファイル名が付いている可能性は高い
    - 同じ名前を活かすと最終的にアップした人のファイルだけ保持される（ほかは上書きされてしまう）
    - 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

ファイルのアップロード

- PHPではファイルのアップロードサイズに制限が設けられている
  - 設定によっても異なる（php.iniなどで設定する）
  - デフォルトは結構小さいファイルサイズのものしかアップロードできない
- アップロードがファイルサイズなどの問題でそもそもできないケースにも対応しておく

問題無くアップロードできれば、
UPLOAD_ERR_OKが入ってくる

```
$_FILES["image"]["error"] !== UPLOAD_ERR_OK
```

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

- 画像はそのまま表示に利用することも多いため、拡張子以外にファイルの中身を確認して形式をチェックした方が良い

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
<?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）";
}
?>
```

指定したファイルが存在するかどうか判別する

画像ファイルの保存

- ファイルがアップロードされてきた場合
  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
  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てきた場合
  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

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
- メールは意外と面倒
  - 文字コードがUTF-8で今はOKなはずだが、古いメーラーなどが対応していない
    （古いのはISO-2022-JP。半角カタカナや絵文字が使えない）
  - 古いメーラーなどにも対応するためには文字コードの変換が必要
  - HTMLメールはもっと面倒

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、バグが発生しづらい
- MAMPではメールを送信するための仕組みが組み込まれていない
  - mail関数は使えるがエラーになる
  - レンタルサーバーなどでは大体利用可能なので、実環境で試す

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
```

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');
?>
```

リダイレクトをする理由

- サーバーサイドではリダイレクトを行うことがある
  - 送信されてきたデータにエラーがあった場合、元のページに戻す
  - ページ内で処理を行い、完了したので、完了ページに移動させる
- ユーザーに対して表示する内容は無いが、処理を行いたい場合に利用する
  - リダイレクト処理を行う場合は、画面に何か表示してしまうとリダイレクトされないので注意

header関数の別利用

- フォームを通じてアップロードされたファイルを出力したい場合
  - 例：JPEG画像をアップロードして貰って、保存した後、PHPからJPEG画像を出力する場合
    - JPEG画像を出力する、とheader関数で指定しないと文字列がブラウザーには表示される
    - ブラウザーはデフォルトでは文字列として全てのデータを受け取って解釈するため
    - header関数でJPEG画像であることをブラウザーに教える

```
header('Content-type: image/jpeg');
```

image/jpegの部分はファイル形式に合わせて変更する

ファイル保存

- 書き込みが可能なフォルダーに対して、PHPからファイルの保存、読み取りが可能
  - WindowsやmacではあまりPermissionを意識しなくて問題無いが、レンタルサーバーなどではパーミッション（書き込みや読み取りの権限）が書き込み可能になっていることを確認すること
- どういう形式のファイルでも保存、読み取りは可能
  - そのままダウンロードさせる、なども可能
  - フォームでアップロードされてきた任意の形式のファイルを保存することも可能

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
     - 開き方の指定次第で指定したファイルが無いとエラーになるパターンと自動的にファイルを作成してくれるパターンが存在する
     - 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（そのまま後に書き足していくモード）で開かれる（一番楽）

テキストファイルの保存

- テキスト情報を保存する場合
  3. 書き込む内容を生成する
     - テキストの場合は、CSV書式（カンマ区切りで値を記述していく）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
  4. 実際に書き込むには、fwrite()関数を利用
     - 第一引数にfopen()関数の返り値を指定、第二引数に書き込む内容を指定する

テキストファイルの読み出し

- 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

- file_get_contents(ファイル名)で読み出すことも可能
  - こちらはファイルを一気に全て読み込む
  - 画像ファイルなどのバイナリファイルの読み出しに適している
  - 外部URLを指定することも可能

ファイルの保存

```
//ファイルへの書き込み
  $fp = fopen('data/data.txt','a');//data/data.txtというファイルを追記モード（ファイルが無ければ作成）で開く
  $content = $name.','.$email.','.$gender.','.$comment."¥n";//書き込む内容を作成する。カンマでそれぞれを区切っておく
  if(fwrite($fp,$content)){
    //fwrite()→指定したファイルに指定した内容を書き込む。成功するとtrueが返る
    print('登録しました');
  }
  else{
    //書き込みに失敗した場合はfwrite()でfalseが返る
    print('書き込みに失敗しました');
  }
```

ファイルの読み出し

```
//ファイルの読み出し
 $fp = fopen('data/data.txt','r');//data/data.txtというファイルを読み出しモードで開く
 while(!feof($fp)){//ファイルの最後まで読み込むと終了
  $line = fgets($fp);//1行分だけ読み出す（勝手に次の行を読んでくれる）
  print($line);//読み出した内容を表示する
 }
```

```
//ファイル全体を一気に読み込む
 $file = file_get_contents('data/imge.jpg'); //data/image.jpgというファイルを全部読み出して$fileに格納する
 header('Content-type: image/jpeg');//jpegファイルを出力することをブラウザーに伝える
 print($file);//ファイル内容を出力
```

ありがとうございました。

また次回。