



G's ACADEMY
TOKYO

追加授業

PHP FileUpload Ajax(axios)



本日専用のファイルとDBを用意します。

本日の配布ファイルは完成形に近いサンプルです。

今日の環境準備

1. DBを構築 → gs_db5
2. サンプル[php05/SQL/gs_db5.sql]をインポートします。
3. htdocsに サンプル [php05] を配置

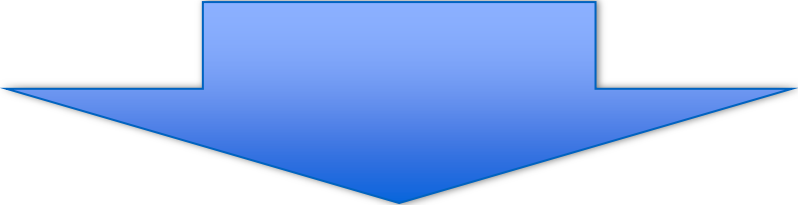
※funcs.phpをhtdocsの上に置くことでブラウザからアクセスできないファイルとします。

Ajaxとは

Ajaxとは

- AjaxのAである「Asynchronous（非同期）」は、非同期でのクライアント・サーバ間の通信を指します
- JavaScriptによりクライアント上での動作が主で必要なデータのみ受信することで通信量の負担を軽減
- 特殊なサーバの設定等は必要としない

Ajaxのメリット

- Webページのリンクをクリックした時のレスポンス待ち時間の体感時間が少ない。
- 必要な部分の情報のみを取得変更し、必要なときに更新可能のため高速に動作する。
- 例)
5画面 → 5HTMLファイル (通常の方法)

5画面 → 1HTMLファイル (Ajaxだとこうできる！)

Ajaxのデメリット

- SEO対策には不向き
- スクリプトの知識が必要
- 作りが複雑になりがち
- 更新履歴が残りません、ブラウザの[戻る]ボタンでは1つ前の状態には戻りません。
- 更新後でも再表示すると初期状態に戻ってしまいます。

Ajaxを使う準備

◇ CDN : HTML内の「 axios 」を使う前に記述しましょう！！

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

【重要】

Ajaxは「 HTTP 」通信で動作します！！

開いてるブラウザの**URLの最初が「 http://... 」 or 「 https://... 」**

どちらかで始まっている必要があります！！

※URLの最初が「 file:///.... 」はNGです！！

```
axios.get('ajax.php').then(function (response) {  
  console.log(response.data); //通信OK  
}).catch(function (error) {  
  console.log(error); //通信Error  
}).then(function () {  
  console.log("Last");//通信OK/Error後に処理を必ずさせたい場合  
});
```

axios基本2[get]

ajax_get.html

//送信データを用意

```
const data = {  
  id:'value1', mode:"value2", type:"value3"  
};
```

//Ajax (非同期通信)

```
axios.get('ajax_get.php' , {  
  params: data  
}).then(function (response) {  
  console.log(response.data); //通信OK  
  document.querySelector("#status").innerHTML="ajax_get.php/通信OK";  
}).catch(function (error) {  
  console.log(error);          //通信Error  
}).then(function () {  
  console.log("Last");          //通信OK/Error後に処理を必ずさせたい場合  
});
```

GET送信 されてる?? <確認する方法>

ajax_get.php/
通信OK

The screenshot shows the Chrome DevTools Network tab. The address bar displays `localhost/php03/ajax/ajax_get.html`. The Network tab is selected, and the filter is set to 'XHR'. A single request is listed with the name `ajax_get.php?id=value1&mode=value2&type=value3`. The request is highlighted, and the 'Headers' sub-tab is active, showing the following headers:

- Sec-Fetch-Mode:** cors
- Sec-Fetch-Site:** same-origin
- User-Agent:** Mozilla/5.0 (Windows NT 10.0; bKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116)

Below the headers, the 'Query String Parameters' are expanded, showing:

- id:** value1
- mode:** value2
- type:** value3

Red dashed boxes and arrows highlight the 'Network' tab, the request name, the 'Headers' sub-tab, and the 'Query String Parameters' section.

axios基本3[post]

ajax_post.html

//送信データをparam変数にセット

```
const params = new URLSearchParams();
```

```
params.append('id', 'value1'); //params.append("パラメータ名", "値");
```

```
params.append('mode', 'value2'); //params.append("パラメータ名", "値");
```

```
params.append('type', 'value3'); //params.append("パラメータ名", "値");
```

//Ajax (非同期通信)

```
axios.post('ajax.php', params).then(function (response) {
```

```
    console.log( response.data ); //通信OK
```

```
}).catch(function (error) {
```

```
    console.log(error); //通信Error
```

```
}).then(function () {
```

```
    console.log("Last"); //通信OK/Error後に処理を必ずさせたい場合
```

```
});
```

axios基本4[post]

ajax_post_cookie.html

//axiosでCookie情報が必要な場合

```
const axiosPost = axios.create({  
  xsrfHeaderName: 'X-CSRF-Token',  
  withCredentials: true  
});
```

//送信データをparam変数にセット

```
const params = new URLSearchParams();  
params.append('id', 'value1'); //params.append("パラメータ名", "値");  
params.append('mode', 'value2'); //params.append("パラメータ名", "値");  
params.append('type', 'value3'); //params.append("パラメータ名", "値");
```

//Ajax (非同期通信)

```
axiosPost.post('ajax.php', params).then(function (response) {  
  console.log( response.data ); //通信OK  
}).catch(function (error) {  
  console.log(error); //通信Error  
}).then(function () {  
  console.log("Last");//通信OK/Error後に処理を必ずさせたい場合  
});
```

POST送信 されてる?? <確認する方法>

ajax_post.php/
通信OK

The screenshot shows a web browser at the URL `localhost/php03/ajax/ajax_post.html`. The Network tab is active, displaying a list of requests. The request `ajax_post.php` is selected, and its details are expanded. The headers section shows `Sec-Fetch-Mode: cors` and `Sec-Fetch-Site: same-origin`. The form data section shows `id: value1`, `mode: value2`, and `type: value3`. Red dashed boxes and arrows highlight the Network tab, the selected request, the Headers section, and the Form Data section.

Network

Filter

All | XHR JS CSS Img Media Font Doc WS Manifest Other

Blocked Requests

100 ms 200 ms 300 ms 400 ms 500 ms

Name

ajax_post.php

axios.min.js

Headers

Sec-Fetch-Mode: cors

Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win bKit/537.36 (KHTML, like Gecko) Chrome/83.0.4 37.36

Form Data

view source view URL encoded

id: value1

mode: value2

type: value3

1 / 4 requests 468 B /



G's ACADEMY
TOKYO

Fileupload



Form ▪ Camera

◇ Camera/写真選択

```
<form method="post" action="送信先" enctype="multipart/form-data">  
  <input type="file" accept="image/*" capture="camera" name="upfile">  
  <input type="submit" value="Fileアップロード">  
</form>
```

① FILE選択できるようにする

```
<input type="file" .....>
```

※他の例<input type="text">

② File送信時はenctype属性を指定

```
enctype="multipart/form-data"
```

③ POST送信 (Action="送信先")

◇ Camera/写真選択

```
<form method="post" action="送信先" enctype="multipart/form-data">  
  <input type="file" accept="image/*" capture="camera" name="upfile">  
  <input type="submit" value="Fileアップロード">  
</form>
```

② カメラ起動 & 画像選択可能

input accept="image/*" capture="camera"

※ 他の記述方法例) accept="image/jpeg, image/gif, image/png"

※ 他の記述方法例) accept="audio/*"

※ 他の記述方法例) accept="video/*"

※ 他の記述方法例) accept="text/comma-separated-values" //CSV

PHP:FileUpload

FileUpload : ①アップロードチェックの処理

◇ \$_FILES : パラメータチェック

ファイルアップロード パラメータ取得

- isset(パラメータ名) (パラメータがセットされているか? ※未入力チェックとは違う)
- \$_FILES["upfile"]["error"] == 0 (0は正常にアップロードしてることを意味する)

※参考URL : http://www.flatflag.nir87.com/move_uploaded_file-970#tablepress-21

```
if( isset( $_FILES["upfile"] ) && $_FILES["upfile"]["error"]==0 ) {  
    // echo 'アップロードしてきている';  
} else {  
    // echo 'アップロードしてきてない OR なにかしらのErrorが発生';  
}
```

◇ \$_FILES : ファイル名、アップロード先のPath取得

```
if( isset($_FILES["upfile"]) && $_FILES["upfile"]["error"]==0 ) {  
    $file_name = $_FILES["upfile"]["name"]; // "1.jpg" ファイル名取得  
    $tmp_path = $_FILES["upfile"]["tmp_name"]; // www/tmp/1.jpg : TempフォルダPath取得  
} else {  
    // echo 'アップロードしてきてない OR なにかしらのErrorが発生';  
}
```

FileUpload : アップロード時のファイル名 重複問題対応

◇ ファイル名重複問題の解決策の例

```
2  if (isset($_FILES["upfile"]) && $_FILES["upfile"]["error"] == 0) {
3      $file_name = $_FILES["upfile"]["name"]; // "1.jpg" ファイル名取得
4      $tmp_path = $_FILES["upfile"]["tmp_name"]; // "/usr/www/tmp/1.jpg" アップロード一時ファイルパス
5      ...
6      /***File名の変更***/
7      $extension = pathinfo($file_name, PATHINFO_EXTENSION);
8      $file_name = date("YmdHis").md5(session_id()).".".$extension;
9
10     $file_dir_path = "upload/".$file_name; // 画像ファイル保管先
11     $img="";
```

◆ 以下 2 行で対応を実現 !

```
/***File名の変更***/
$extension = pathinfo($file_name, PATHINFO_EXTENSION);
$file_name = date("YmdHis").md5(session_id()).".".$extension;
```

FileUpload:②アップロード処理

◇ アップロードに使用する関数: 3ステップ!

1. `is_uploaded_file` アップロードOK!?
2. `move_uploaded_file` Tempフォルダからimgフォルダへ移動
3. `chmod` ファイルに権限を付与する (ブラウザで見れるように)

```
// FileUpload [--Start--]
if ( is_uploaded_file( $tmp_path ) ) {
    if ( move_uploaded_file( $tmp_path, $file_dir_path ) ) {
        chmod( $file_dir_path, 0644 );

    } else {
        // echo "Error:アップロードできませんでした。";
    }
}
// FileUpload [--End--]
```


FileUpload : ③アップロード完成例

◇ fileupload処理の流れ（シンプルバージョン）

```
<?php
```

```
if ( isset($_FILES["upfile"]) && $_FILES["upfile"]["error"]==0 ) {  
    $file_name = $_FILES["upfile"]["name"];          //"1.jpg"ファイル名取得  
    $tmp_path = $_FILES["upfile"]["tmp_name"];        //"../www/tmp/1.jpg"などの一時フォルダ  
  
    //ユニークファイル名作成  
    $extension = pathinfo($file_name, PATHINFO_EXTENSION);  
    $file_name = date("YmdHis").md5(session_id()) . "." . $extension;  
    $file_dir_path = "upload/" . $file_name;          //画像ファイル移動先とファイル名  
  
    // FileUpload [--Start--]  
    if ( is_uploaded_file( $tmp_path ) ) {  
        if ( move_uploaded_file( $tmp_path, $file_dir_path ) ) {  
            chmod( $file_dir_path, 0644 ); //ファイルアップ完了&アクセス権限付与  
            $ret = $file_name;  
        } else {  
            $ret = "Error:アップロードできませんでした。"; //Error文字  
        }  
    }  
} else {  
    $ret = "画像が送信されていません"; //Error文字  
}
```

実際のファイルに追加

今まで作ってきたファイルに追加です！
しかし、その前に関数化！

FileUpload関数化

```
43 //fileUpload("送信名","アップロード先フォルダ");
44 ▼ function fileUpload($fname,$path){
45 ▼ ... if (isset($_FILES[$fname]) && $_FILES[$fname]["error"] == 0) {
46     ... //ファイル名取得
47     ... $file_name = $_FILES[$fname]["name"];
48     ... //一時保存場所取得
49     ... $tmp_path = $_FILES[$fname]["tmp_name"];
50     ... //拡張子取得
51     ... $extension = pathinfo($file_name, PATHINFO_EXTENSION);
52     ... //ユニークファイル名作成
53     ... $file_name = date("YmdHis").md5(session_id()).".".$extension;
54     ... //FileUpload[--Start--]
55     ... $file_dir_path = $path.$file_name;
56 ▼ ... if (is_uploaded_file($tmp_path)) {
57 ▼ ... if (move_uploaded_file($tmp_path,$file_dir_path)) {
58     ... chmod($file_dir_path, 0644);
59     ... return $file_name; //成功時：ファイル名を返す
60 ▼ ... } else {
61     ... return 1; //失敗時：ファイル移動に失敗
62     ... }
63     ... }
64 ▼ ... } else {
65     ... return 2; //失敗時：ファイル取得エラー
66     ... }
67 }
```

FileUpload関数 使用方法

```
2 //関数読み込み
3 include("../funcs.php");
4
5 //ファイルアップロード処理
6 $status = fileUpload("upfile", "upload/"); //戻り値 : 0=ファイル名, 1=NG, 2=NG
7 if($status==1 || $status==2){
8     ...$img = "アップロード失敗";
9 }else{
10     ...$img = $status; //ファイル名
11 }
12 ?>
13
```

画像をDBへ保存

insert.php の例

- DBに保存するのはファイル名のみ(DBが重くなるから！)
- ファイルアップロード処理の下でSQLを作成しましょう！
例) `INSERT INTO ****(...., img, ...) VALUES(..., :img, ...)`
- bindValueを作成！
例) `$stmt->bindValue(":img", "ファイル名が入ってる変数");`

以上

画像表示方法

select.php の例

表示方法 例)

```
">
```

上記は、imgカラムに画像ファイル名が保存されてる
場合の例で、\$rにデータが入ってる想定です。

パスワードのhash化

既にしてあります！

(login_act.php , user_insert.php)

パスワード比較 パスワードハッシュ化

～ 2つの関数のみ使用 ～

◇パスワードハッシュ作成

`password_hash("登録する文字", PASSWORD_DEFAULT);`

※ DB:Passwordカラム型をvarchar(255)に変更!

※ ユーザー登録時に使用 (ハッシュ化してDBに登録しておくため)

<http://php.net/manual/ja/function.password-hash.php>

◇パスワードのマッチチェック


`password_verify("パスワード入力値", "DB値");`

※ LOGIN認証時に使用する (ハッシュ文字と入力文字を比較して判定する関数)

<http://php.net/manual/ja/function.password-verify.php>

Passwordハッシュ化：事前準備

データベース >> gs_user_table >> lpwカラムを変更
varchar(255) に！



サーバ: localhost » データベース: gs_db » テーブル: gs_user_table

表示 構造 SQL 検索 挿入 エクスポート


名前	データ型 ?	長さ/値 ?	デフ:
lpw	VARCHAR	255	な

1. テストデータのパスワードをハッシュ化

```
11 $name      = filter_input( INPUT_POST, "name" );
12 $lid       = filter_input( INPUT_POST, "lid" );
13 $lpw       = filter_input( INPUT_POST, "lpw" );
14 $kanri_flg = filter_input( INPUT_POST, "kanri_flg" );
15 $lpw       = password_hash($lpw, PASSWORD_DEFAULT); //パスワード
16
① 17 //2. DB接続します
18 $pdo = db_conn();
19
20 //3. データ登録SQL作成
21 $sql = "INSERT INTO gs_user_table(name,lid,lpw,kanri_flg,life_
22 $stmt = $pdo->prepare($sql);
23 $stmt->bindValue(':name', $name, PDO::PARAM_STR); //Integer (数
24 $stmt->bindValue(':lid', $lid, PDO::PARAM_STR); //Integer (数値
25 $stmt->bindValue(':lpw', $lpw, PDO::PARAM_STR); //Integer (数値
26 $stmt->bindValue(':kanri_flg', $kanri_flg, PDO::PARAM_INT); //
27 $status = $stmt->execute();
```

① user_insert.php (ユーザー登録時にパスワードをハッシュ化 password_hash関数)

② ハッシュ化してるデータを通常通り登録



id	name	lid	lpw
1	TEST	test	\$2y\$10\$jmot7MkoGd4R7Z0bNCFwVeTWweYUWOPViFc9N1vqtNydyrLPIX3Cu

ハッシュ化する前はtest

user登録処理に"password_hash()"を使いパスワードをハッシュ化して登録させます。

2. "login_act.php"の一部を修正

①SQLとbindValueを修正

```
$sql = "SELECT * FROM gs_user_table WHERE lid=:id";  
$stmt = $pdo->prepare($sql);  
$stmt->bindValue(':id', $lid);  
$res = $stmt->execute();
```

②password_verify関数を使ってパスワードを比較

```
if( password_verify($lpw, $val["lpw"]) ){  
    $_SESSION["chk_ssid"] = session_id();  
    $_SESSION["kanri_flg"] = $val['kanri_flg'];  
    $_SESSION["name"] = $val['name'];  
    header("Location: select.php");  
}else{  
    //logout処理を経由して全画面へ  
    header("Location: login.php");  
}
```