

「Web プログラミング演習Ⅱ A/B」 まとめ

学習内容：PHPを使ったWebアプリケーションを作成する。

テキスト：確かな力が身につくPHP「超」入門（SBCreative）

0. 予備知識

PHPの学習前に、HTMLやCSS、SQLの基本内容を事前学習しておいた方がよい

今回の対象学生は、前年度にJava言語を学習しており、プログラミングの基本は習得していた

■HTML (Hyper Text Markup Language)

Webページを作るときに使われるマークアップ言語

Webページの文書構造を定義し、タグを使って記述する

利用者の立場に立って分かりやすく表示するように心掛ける

■HTML5.1 主要タグ

<html> 文書のルート要素

文書のメタデータ

<head>	メタデータの集まりを表す
<title>	文書にタイトルを付ける
<link>	別の文書類と関連付ける
<style>	スタイルシートを指定する
<meta>	文書に関する様々なメタデータを表す

セクション

<body>	文書の本体を表す
<article>	自己完結したセクションを表す
<section>	一般的なセクションを表す
<nav>	ナビゲーションを表す
<aside>	補足的なセクションを表す
<h1>-<h6>	見出しを付ける

コンテンツのグループ化

<p>	段落を表す
<hr>	テーマの区切りを表す
	順序付きのリストを作る
	順不同のリストを作る
	リストの項目を表す
<dl>	記述リストを作る
<dt>	記述リストの名前部分を表す
<dd>	記述リストの値部分を表す
<div>	特定の範囲をグループ化する

テキストの意味

<a>	リンクを設定する
-----	----------

	強調を表す
	重要性を表す
	特定の範囲をグループ化する
 	改行する

コンテンツの埋め込み

	画像を表示する
<iframe>	インラインフレームを作る
<object>	外部リソースを埋め込む

テーブル

<table>	表を作る
<tr>	表の行を表す
<td>	データセルを表す
<th>	見出しセルを表す

フォーム

<form>	入力フォームを作る
<label>	フォーム部品とラベルを関連付ける
<input>	入力欄・選択肢・ボタンを作る
<select>	プルダウンメニューを作る
<option>	プルダウンメニューや入力候補の選択肢を作る
<textarea>	複数行のテキスト入力欄を作る
<button>	内容を持つボタンを作る

スクリプティング

<script>	JavaScript を埋め込む
<canvas>	スクリプト使ってグラフィックスを描く

■ C S S (Cascading Style Sheets)

Web ページのスタイルを指定するための言語で、HTML と組み合わせて使用する

文字や画像、表やレイアウトなどの見栄えをよくする

外部 CSS ファイルを呼び出す方法と文書単位に適用する方法があるが、前者が一般的である

■ S Q L (Structured Query Language)

リレーショナルデータベースの データを効率よく操作するための言語

・ SELECT (選択表示)

SELECT <列名 1>, <列名 2>, … FROM <テーブル名> WHERE <条件>;

・ INSERT (追加)

INSERT INTO <テーブル名> (列 1, 列 2, 列 3, …) VALUES (値 1, 値 2, 値 3, …);

・ UPDATE (変更)

UPDATE <テーブル名> SET <列名> = <値> WHERE <条件式>;

・ DELETE (削除)

DELETE FROM <テーブル名> WHERE <条件式>;

1. PHPの概要

■PHP とは

1995 年に開発されたスクリプト言語で、Web アプリケーションの開発に向いている
ショッピング、ブログ、SNS、検索・予約サイトなど日常的に使用されている

■PHP が動く仕組み

<クライアントサイド>

(ブラウザ)

①ブラウザから Web サーバへリクエストを送る

<サーバサイド>

(Web サーバ + データベース)

②Web サーバがスクリプトを実行する

③データベースを操作する

④データベース操作の結果が返される

⑤スクリプト実行結果が Web サーバ に返される

⑥Web サーバからブラウザにレスポンスが送られる

■PHP の動かし方

①記述 開始タグ「<?php」と終了タグ「?>」の間にコードを記述する

②保存 拡張子を「.php」で保存する

③実行 ブラウザから「.php」ファイルの URL を開く

■他言語との違い

HTML は内容が固定的だが、PHP は検索結果など動的に生成する

JavaScript はクライアントサイドで実行されるが、PHP はサーバサイドで実行される

Java よりも簡潔に記述できる

2. 環境構築

■XAMPP（ザンプ）

PHP の開発に必要なソフトウェアを一括して簡単に導入できる

1 台の手元コンピュータでブラウザ(クライアント)と Web サーバ(サーバ)を動かすことができる

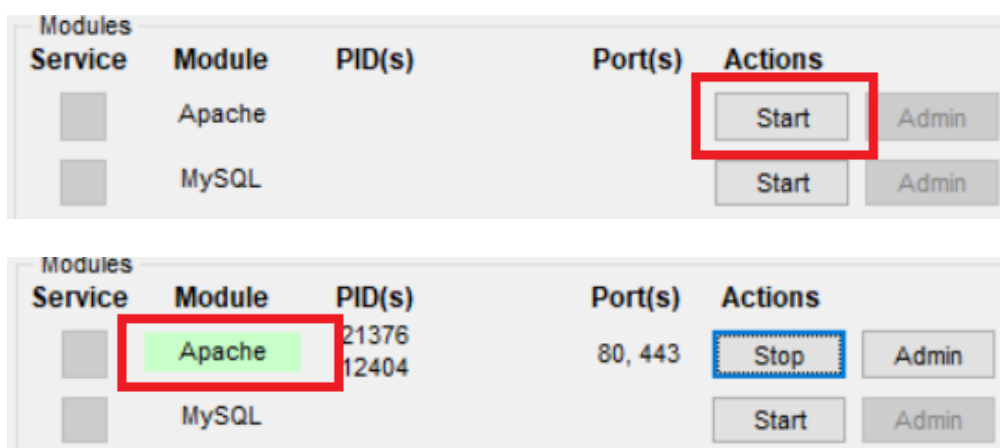
- ・ Apache Web サーバ
- ・ MariaDB MySQL から派生したデータベース管理システム
- ・ PHP PHP で開発を行うために必要な環境
- ・ Perl Perl というプログラミング言語で開発を行うために必要な環境



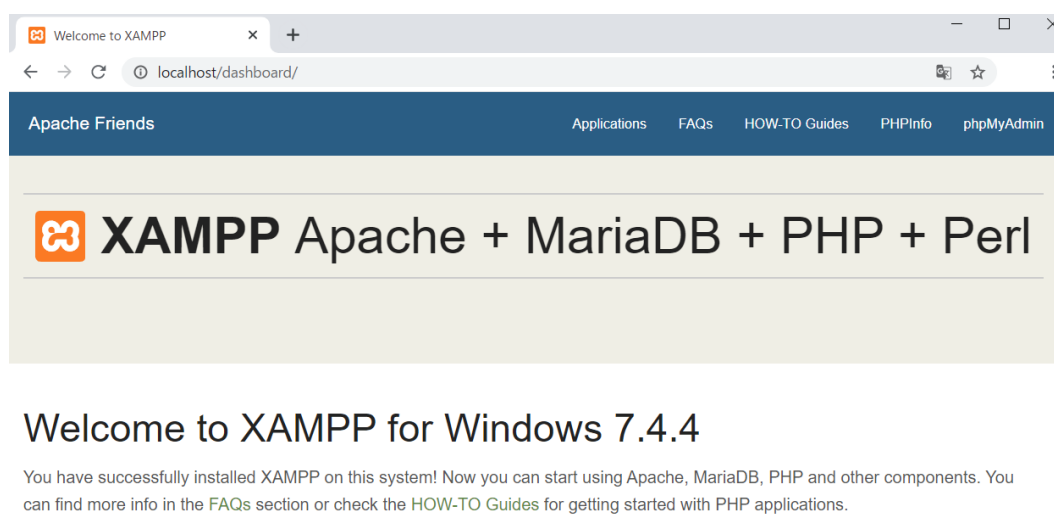
<https://www.apachefriends.org/jp/index.html> からインストール、ダウンロード

■Apache（アパッチ）の起動確認

XAMPP コントロールパネルから Apache の[Start]ボタンで起動させる（緑色）



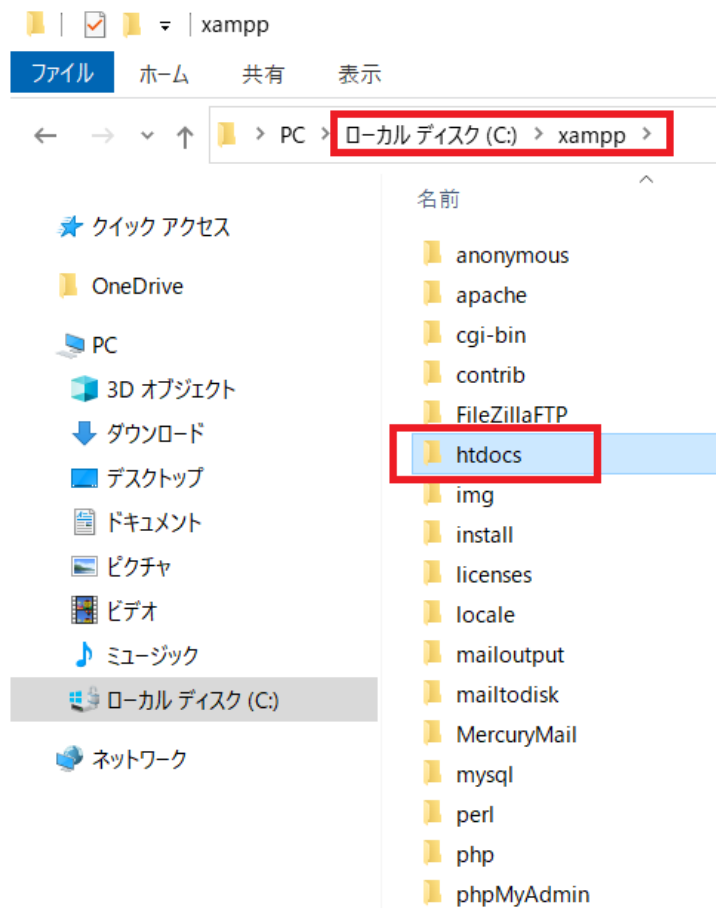
ブラウザに <http://localhost> を入力して開くと、XAMPP の紹介ページが開くことで起動確認
localhost を指定することで、XAMPP の Apache を Web サーバとして利用できる



■スクリプトの実行

XAMPP では、htdocs フォルダ以下に PHP スクリプトや HTML ファイルを配置する

<http://localhost/フォルダ名/ファイル名> は c:\xampp\htdocs\フォルダ名\ファイル名と対応している
PHP スクリプトを実行するには、スクリプトの URL を指定する



■テキストエディタ

メモ帳、TeraPad、秀丸など

ファイル保存時に、文字コードとして UTF-8 が指定できるものにする
文字コードの設定が違っていると文字化けを起こすことがある

■ショートカットキーなど

コピー Ctrl + C

切り取り Ctrl + X

貼り付け Ctrl + V

元に戻す Ctrl + Z

保存 Ctrl + S

範囲指定 Shift + 矢印キー

再読込 F5

3. PHP 基礎

■PHP スクリプト

開始タグ(<?php)と終了タグ(?>)の間にスクリプトを記述する

```
<?php
    PHP スクリプト
    :
?>
```

1つの文の最後に「;」(セミコロン)をつける

文字列を記述する場合は「'」(シングルクォート)または「"」(ダブルクォート)で文字列を囲む

■HTML 文書に PHP を埋め込む

HTML 文書の body タグ内に PHP を記述する

記述例

```
<html>
    <head>
        :
    </head>
    <body>
        HTML ドキュメント
        <?php
            PHP スクリプト
            :
        ?>
        HTML ドキュメント
    </body>
</html>
```

→内容固定 (そのまま出力)

→内容可変 (PHP の実行結果が表示される)

→内容固定 (そのまま出力)

■メッセージ出力 (echo)

構文 : echo '出力したい文字列';

使用例: echo 'Hello World';

文字列結合は「,」(コンマ)または「.»(ピリオド)を使用する

使用例: echo 'Hello',' World';

メッセージ出力には print もあるが、高速性、機能面で echo を使用する場合が多い

「"」はテキスト中に変数の埋め込みができるが、「'」はできない

■変数

数値型や文字型など型の指定や宣言は不要で、すぐに使える

変数名についての規則

- ・変数名の前には「\$」(ドル記号)をつける
- ・1文字目は英字または「_」アンダースコア
- ・2文字目以降は英字、数字、アンダースコアのいずれか

・英字の大文字と小文字は区別される

使用例: `$price`, `$price2`, `$price_tag` など

■定義済みの変数

`$_POST` HTTP の POST リクエストパラメータ
`$_GET` HTTP の GET リクエストパラメータ
`$_REQUEST` HTTP のリクエストパラメーター式 (POST および GET)
`$_SESSION` セッション
`$_COOKIE` クッキー
`$_FILES` アップロードされたファイルの情報

■変数への代入・参照

代入例 1 `$team = 'ジャイアンツ';`

参照例 1 `echo $team;` // 「ジャイアンツ」表示

代入例 2 `$count = 1;`

`$count = $count + 2;`

参照例 2 `echo $count;` // 「3」表示

■定数

構文 `const 定数名 = 値;`

使用例 `const TAX = 10;` `const MESSAGE = 'ありがとう';`

定数の先頭に「\$」はつけない

定数名の命名規則は変数と同じだが、大文字を使う習慣がある

■演算子

`+`, `-`, `*`, `/` (加減乗除) `**` (累乗) `%` (剰余)

`++` (インクリメント) `--` (デクリメント)

`=` (代入) `==` (等しい) `<`, `<=`, `>`, `>=` (比較)

`&&` (かつ) `||` (または) `!` (否定)

■ファイルの呼び出しと実行 (require 文)

使いまわす部分を別ファイルに保存しておき、require 文で読み込み実行させる

同じ内容を複数個所入力する手間が省ける

変更する必要が出てきたとき、変更するファイルが少なくなる

構文 `require 'ファイル名';` ※ファイル名のパスは相対パスが一般的

使用例 `require '../header.php';`

文書のヘッダーやフッターなど複数のスクリプトで使いまわす処理をまとめておくと良い

■ヘッダーとフッターの保存例

```

<html>
  <head>
    <meta charset="UTF-8">
    <title> PHP Sample </title>
    <link rel="stylesheet" href="../style.css">
  </head>
  <body>
    <?php
      PHP スクリプト
      :
    ?>
  </body>
</html>

```

header.php に保存

footer.php に保存

■リクエストパラメータ

Web サーバに対してリクエストを行う際に、同時に送信する付加的な情報のこと
 入力されたデータを Web サーバに送信して処理させるときに必要

記法 \$_POST[‘リクエストパラメータ名’]

コントロールの name 属性の値がリクエストパラメータ名となる

■POST と GET

POST でデータを送ると、表面上そのデータを隠ぺいできる

機密情報のやりとりや大量のデータをはじめ、通常は POST で送信する

GET でデータを送ると、URL にその内容が表示される（URL の最後に ?データ名=値）

簡単な情報のやりとりのみ使用される

■テキストボックス

テキストを入力するためのコントロール

構文 <input type="text" name="名前">

パスワードは type="password" とする（入力すると●●など表示される）

■ボタン

Web サーバにフォームの内容を送信するためのコントロール

構文 <input type="submit" value="表示文字">

■フォーム送信時の動作

フォーム上のボタンを押すと、コントロールの値が出力用スクリプトに送信される

<form>から</form>までの間にフォームで使用するコントロールを記述する

構文例 <form action="出力用スクリプト" method="POST">

 コントロール（テキストボックスなど）

 コントロール（ボタン）

 </form>

4. 制御構造とコントロール

■条件分岐 (if 文)

```
構文    if(条件式 1){
        // 条件式 1 を満たした場合に実行される処理
    }elseif(条件式 2){
        // 条件式 1 を満たさず、条件式 2 を満たした場合に実行される処理
    }else{
        // 条件式 1、2 を満たさなかった場合に実行される処理
    }
```

■条件分岐 (switch 文)

```
構文    switch(変数) {
        case 値 1:           //変数が値 1 の時、処理 A を実行する
            処理 A
            break;
        case 値 2:           //変数が値 2 の時、処理 B を実行する
            処理 B
            break;
        default:
            処理 X           //どれにもあてはまらない時、処理 X を実行する
    }
```

各処理後に break を入れないと、次の case も実行される

■ループ (for 文)

```
構文    for(初期値; 継続条件; 増分値) {
        繰り返す処理
    }
```

一定回数の繰り返しの時に使用する

■ループ (while 文)

```
構文    while(継続条件) {
        繰り返す処理
    }
```

何回繰り返すか分からないときに使用する

■配列

変数は1つだけの値を格納するが、配列は複数の値を格納することができる

添字には、整数（ゼロから）または文字列を使うことができる

構文 (格納) \$配列名 = [値1, 値2, 値3…];

記法 (参照) \$配列名[添字]

■連想配列

キーと値の組み合わせを保存できるようにした配列で、PHP で使用できる

構文 (格納) \$配列名 = [キ-1=>値1, キ-2=>値2, キ-3=>値3, …];

使用例 `$bango=['福岡'=>40,'佐賀'=>41,'長崎'=>42];`

■配列の要素の追加、上書き、削除のやり方

構文 (追加) \$array[] = 追加したい要素;

構文（上書き） `$array[上書きしたい要素のキー番号] = 上書きしたい内容;`

構文 (削除) `unset($array[削除したい要素のキー番号]);`

※削除した場所はつまりず、そのまま空く

■ループ (foreach 文)

配列などを参照しながら、データがある間ループしてくれるので便利

1 回ごとに配列の中身を取り出しながら処理を繰り返すことができる

```
構文1  foreach(配列 as 変数){
        変数を使った処理;
    }
```

```
使用例 1 $week=['月','火','水','木','金','土','日'];
foreach($week as $day){ //配列$week から 1 つずつ$day に取り出して繰り返す
    echo $day;
} //実行すると、月火水木金土日と表示される
```

■ループ（連想配列の foreach 文）

```
構文 2    foreach(配列 as キーの変数 => 値の変数) {
            キーの変数と値の変数を使った処理;
        }
```

```
使用例 2 $bango=[ '福岡'=>40,'佐賀'=>41,'長崎'=>42];
foreach($bango as $key=>$value){ //キーは$key に、値は$value に記憶される
    echo $key,'は',$value,'番です<br>'; //1 回目は 福岡は 40 番です と表示
}
```

■チェックボックス

ユーザーにオンまたはオフを選択してもらうコントロール

構文 `<input type="checkbox" name="名前">`

isset 関数でチェックされているかどうか判定できる

if 文や foreach 文で判定処理することが多い

■ラジオボタン

グループの中から 1 つだけを選択するコントロール

構文 `<input type="radio" name="名前" value="値">`

name 属性の名前が同じラジオボタンは 1 つのグループとなる

あらかじめ選択状態にするには<input>タグに checked と記述する

選択した radio ボタンの value 属性の値が受け渡される値となる

switch 文で判定処理することが多い

■セレクトボックス

複数の項目の中から 1 つを選択するためのコントロール

ドロップダウンメニューとも呼ばれる

構文 `<select name="名前">`

`<option value="値 1">表示文字 1 </option>`

`<option value="値 2">表示文字 2 </option>`

:

`</select>`

選択された項目の value 属性の値が受け渡される値になる

switch 文で判定処理することが多い

5. 関数ほか

■関数の定義

関数とは、1つのまとまった処理を必要な時に呼び出せるようにしたもの
関数は引数を渡して実行し、実行結果は戻り値として呼び出し元に渡される

```
構文(定義)    function 関数名 (引数) {
                実行する処理;
                return 戻り値      ←戻り値なしの場合は不要
            }
```

構文(実行) 関数名 (引数) ;

■date_default_timezone_set 関数

取得したい地域に合わせてタイムゾーンを指定する

使用例 date_default_timezone_set('Japan');

■date 関数

現在の日付を指定した形式で表示する

使用例 echo date('Y/m/d H:i:s'); → 2020/12/24 のように表示

■rand 関数

最小値以上、最大値以下の乱数を生成する

使用例 echo rand(1,6); → 1,2,3,4,5,6 のいずれかが表示

■preg_match 関数

正規表現によるパターンマッチングを行う

構文 preg_match('/パターン/', 入力文字列)

マッチすると「1」(TRUE)、マッチしなかった場合は「0」(FALSE)を返す

使用例 1 if(preg_match('/^[0-9]{3}-[0-9]{4}\$/', \$postcode)){
 「^」 行頭 ↑ 郵便番号 (ハイフンあり)
 「[0-9]」 0 から 9 までの数字 1 文字
 「{3}」 直前の文字が 3 文字
 「\$」 行末

使用例 2 preg_match('/(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]{8,}/', \$password))
 → 8 文字以上で、英小文字、英大文字、数字を各 1 文字以上含むパスワード
 「[a-z]」 英小文字、「[A-Z]」 英大文字、「[a-zA-Z]」 英小文字・英大文字のいずれか
 「.*」 任意の文字繰り返し、 「?=」 以降の文字を含む、「{8,}」 8 文字以上

■mb_convert_kana 関数

半角と全角を変換する

入力されたデータを統一した形で保存することで処理エラーを防ぐことができる

構文 mb_convert_kana(文字列, オプション)

オプション例 r(全角英字を半角英字に変換) R(半角英字を全角英字に変換)

n(全角数字を半角数字に変換) N(半角数字を全角数字に変換) など

■file_exists 関数

指定のファイルまたはフォルダが存在するかどうか調べる

構文 file_exists(ファイル名またはフォルダ名)

指定したものが存在する場合には TRUE、存在しない場合には FALSE を返す

■file_get_contents 関数

ファイルの全体を読み込み、内容を文字列として返す

構文 file_get_contents(ファイル名)

■json_decode 関数

JSON 形式の文字列を解釈して、PHP の文字列や配列といったデータに変換する

構文 json_decode(文字列)

■json_encode 関数

ファイルに保存するための JSON 形式に変換する

構文 json_encode(値)

■file_put_contents 関数

指定した文字列を指定したファイルに書き込む

構文 file_put_contents(ファイル名, 文字列);

■アップロード用のフォーム

form タグの enctype 属性を multipart/form-data、input タグの type 属性を file にする

構文例 <form action="出力用スクリプト" method="POST" enctype="multipart/form-data">
 <input type="file" name="名前">
 </form>

■\$_FILES 変数

アップロードされたファイルの情報を記憶している変数

<form>タグのファイル選択欄からアップロードされたファイルは一時的なファイルに保存

構文 \$_FILES['名前']['tmp_name'] //一時的なファイル名

\$_FILES['名前']['name'] //アップロードされたファイル名

■is_uploaded_file 関数

入力画面からアップロードされたファイルかどうかを調べる

アップロードされたファイルの場合、TRUE を返す

構文 is_uploaded_file(ファイル名)

■mkdir 関数

フォルダを作成する

構文 mkdir(フォルダ名);

■basename 関数

パスの末尾にあるフォルダ名やファイル名だけを取り出す

構文 basename(パス)

■move_uploaded_file 関数

アップロードされた一時的なファイルを保存先のファイルに移動する

構文 `move_uploaded_file(一時的なファイル, 保存先のファイル)`
 移動に成功したときに TRUE を返す

■isset 関数

変数が定義されているかどうかを判定する

使用例 `if(isset($_POST['user'])){`
 定義済みのときの処理
 `} else {`
 未定義のときの処理
 `}`

■empty 関数

値が空のときに TRUE を返す

使用例 `if(empty($_POST['name'])){`

■htmlspecialchars 関数

HTML タグを無効化する（データの安全性を高めるため、クロスサイトスクリプティング対策）

使用例 `echo htmlspecialchars($_POST['user']);`

■print_r 関数

指定した配列、変数に関する情報を解りやすく出力する。

構文 `print_r(配列名);`

■explode 関数

特定の区切り文字で区切られた文字列を分割して配列に代入する

CSV データを扱う際や、URL の分割などに使用する

構文 `$配列名 = explode("分割する区切り文字", 分割したい文字列);`

使用例 `$string = "a/b/c";`
 `$array = explode("/", $string);` // 「/」 で区切られた文字を分割して配列に代入
 // \$array[0]には a が、\$array[1]には b が、\$array[2]には c が代入される

■count() 関数

配列に含まれるすべての要素の数を数える

テーブル名に [] はつけない。削除された空要素は含まない。

テーブル処理のループ条件でよく使う

構文 `count(配列名);`

■クラス

PHP でもクラスを利用したオブジェクト指向開発が可能

共通的な処理はクラス化して再利用していくことが多い

構文例（定義） `class クラス名{`
 `private $変数名; //public や private 等で変数定義`
 :
 `public function メソッド名 ($引数名 1,$引数名 2) {`
 処理;
 `}`

```
    }
}
```

構文例（利用） `require 'ファイル名';`
 `$インスタンス名 = new クラス名;`
 `$インスタンス名 -> メソッド名`

■try/catch

例外処理

構文 `try{`
 `//例外が発生する可能性がある処理`
 `}catch(例外の型 $ex){`
 `//例外が発生したときの処理`
 `}`

catch で例外の種類ごとに処理を切り替えられる

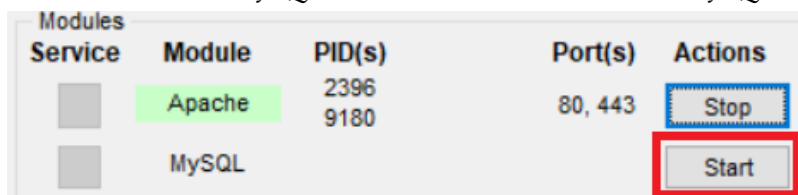
例外にはいろいろな種類がある

プログラム上で不具合となる現象が発生した場合、わざと例外を投げる(throw)ことがある

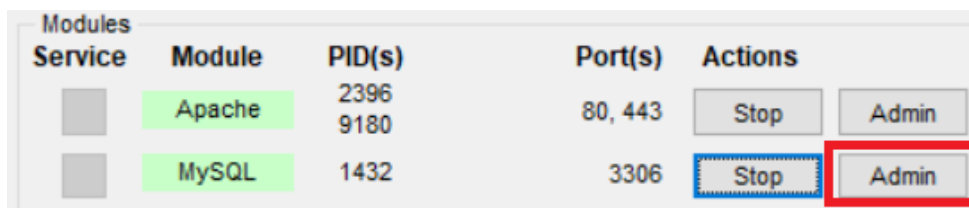
6. データベース基本操作

■データベースの作成（マウス操作等による作成）

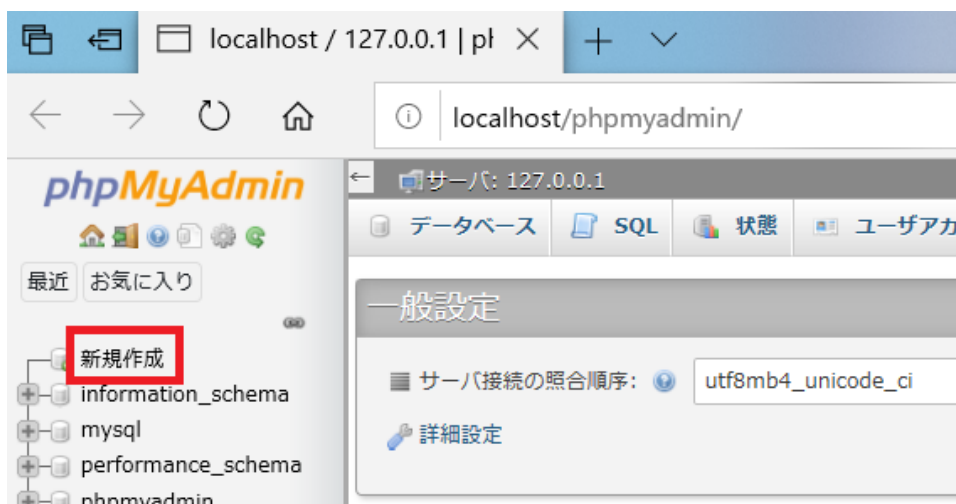
①XAMPP を起動し、MySQL の「Start」をクリックして MySQL を起動



②MySQL の「Admin」をクリックして phpMyAdmin を起動



③「新規作成」をクリックしてデータベースを作成

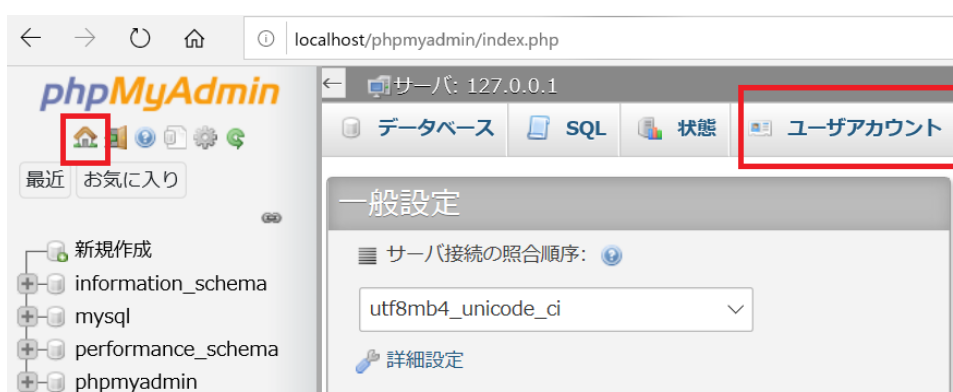


④データベース名（ここでは shop）入力し、照合順序「utf8_general_ci」を選択して「作成」

shop utf8_general_ci 作成

■ユーザの作成

①「メインページ」ボタンから「ユーザアカウント」をクリック



- ②新規作成「ユーザアカウントを追加する」をクリック



- ③ユーザ名、ホスト名、パスワードを入力し、下部の「実行」ボタンをクリック
(ここではユーザ名：staff ,ホスト名：localhost ,パスワード：password で入力)

The image shows the 'ログイン情報' (Login Information) form. It has four rows: 'ユーザ名' (Username) with a dropdown menu, 'ホスト名' (Host) with a dropdown menu, 'パスワード' (Password) with a dropdown menu, and '再入力' (Re-enter). The input fields are highlighted with a red box.

- ④「データベース」をクリック (ここでは「shop」) を選んで「実行」



- ⑤特権を追加したいデータベース (ここでは「shop」) を選んで、「実行」クリック

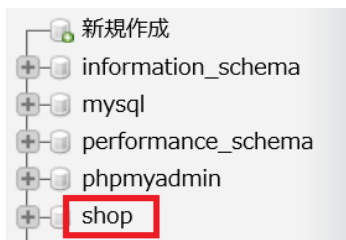


- ⑥「データ」にチェックをして「実行」

The image shows the 'データ' (Data) section. It has a checkbox labeled 'データ' (Data) which is checked. Below it are four checkboxes: 'SELECT', 'INSERT', 'UPDATE', and 'DELETE', all of which are checked.

■テーブルの作成

- ①phpMyAdmin のツリーから、作成したいデータベース（ここでは「shop」）をクリック



- ②「テーブルを作成」に「名前」と「カラム数」（列数）を入力して、「実行」をクリック

テーブルを作成

名前: カラム数:

- ③「名前」（項目名）「データ型」「長さ/値」などを設定し「保存する」をクリック

名前	データ型	長さ/値	デフォルト値	照合順序
<input type="text" value="中央カラムから選択"/>	INT	<input type="text"/>	なし	<input type="text"/>
<input type="text" value="中央カラムから選択"/>	INT	<input type="text"/>	なし	<input type="text"/>
<input type="text" value="中央カラムから選択"/>	INT	<input type="text"/>	なし	<input type="text"/>

■テーブルにデータを追加

- ①phpMyAdmin ツリーのデータベース「shop」を選択、テーブル「product」選択
「挿入」ボタンをクリックし、データを入力する

- ②テーブル内容の確認

product テーブルを選択し、表示タブをクリックするとテーブルの内容が確認できる。

サーバ: 127.0.0.1 データベース: shop テーブル: product

表示 構造 SQL 検索 挿入 その他

行 0 - 10 の表示 (合計 11, クエリの実行時間: 0.0008 秒。)

SELECT * FROM `product`

プロファイリング [インラインを編集する] [編集] [EXPLAIN で確認] [ドの作成]

すべて表示 行数: 25 行フィルタ: このテーブルを核

	id	name	price
<input type="checkbox"/> 編集 コピー 削除	1	松の実	700
<input type="checkbox"/> 編集 コピー 削除	2	くるみ	270
<input type="checkbox"/> 編集 コピー 削除	3	ひまわりの種	210
<input type="checkbox"/> 編集 コピー 削除	4	アーモンド	220
<input type="checkbox"/> 編集 コピー 削除	5	カシューナッツ	250
<input type="checkbox"/> 編集 コピー 削除	6	ジャイアントコーン	180
<input type="checkbox"/> 編集 コピー 削除	7	ピスタチオ	310
<input type="checkbox"/> 編集 コピー 削除	8	マカダミアナッツ	600
<input type="checkbox"/> 編集 コピー 削除	9	かぼちゃの種	180
<input type="checkbox"/> 編集 コピー 削除	10	ピーナッツ	150
<input type="checkbox"/> 編集 コピー 削除	11	クコの実	400

■SQL 文によるデータベース・ユーザ・テーブルの作成

①テキストエディタ等に以下の内容の SQL 文を入力

【product.sql】

drop database if exists shop;

←DB 削除

create database shop default character set utf8 collate utf8_general_ci;

←DB 作成

grant all on shop.* to 'staff'@'localhost' identified by 'password';

←権限付与

use shop;

←DB 接続

create table product (

←テーブル作成

id int auto_increment primary key,

name varchar(200) not null,

price int not null

);

insert into product values(null, '松の実', 700);

←データ追加

insert into product values(null, 'くるみ', 270);

insert into product values(null, 'ひまわりの種', 210);

insert into product values(null, 'アーモンド', 220);

insert into product values(null, 'カシューナッツ', 250);

insert into product values(null, 'ジャイアントコーン', 180);

insert into product values(null, 'ピスタチオ', 310);

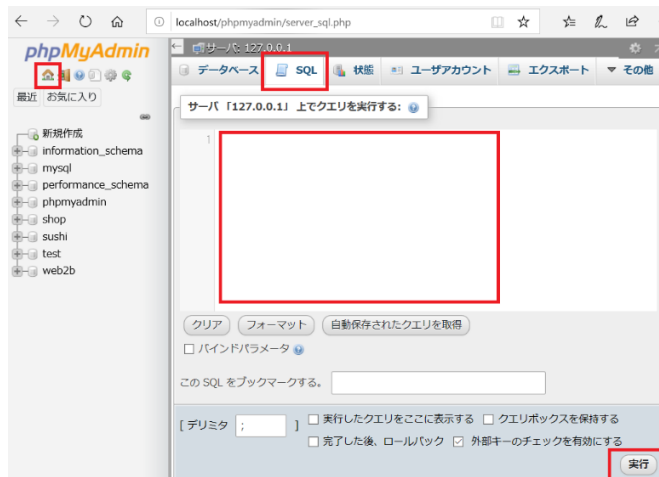
insert into product values(null, 'マカダミアナッツ', 600);

insert into product values(null, 'かぼちゃの種', 180);

insert into product values(null, 'ピーナッツ', 150);

insert into product values(null, 'クコの実', 400);

- ②「SQL」タブから、SQL 入力欄に①のスクリプトをコピーして貼り付け、「実行」クリック



- ③テーブル内容の確認

product テーブルを選択し、SQL タブから下の SQL を入力し実行する
 SELECT * FROM product;

■商品一覧の表示

- ①PDO クラスでデータベースに接続する

PDO(Php Data Object)はクラスの一つで、PHP とデータベースとの間の接続機能を提供
 データベースを操作するための変数や関数がまとめて定義されている

構文例 `$pdo = new PDO('mysql:host=サーバの場所;dbname=DB 名;charset=utf8',
 'ユーザ名','パスワード');`

- ②SQL の生成

使用例 `$sql = "SELECT * FROM produce";`

- ③SQL の実行

PDO クラスの query メソッドを呼び出して実行するするには、「変数->メソッド」と記述

使用例 `$selectdata = $pdo->query($sql);`

- ④実行結果の読み込み

1 行分のデータは配列形式で格納されており、列名を配列の添字に指定する

使用例

```
foreach($selectdata as $row){  
    echo 'ID は'.$row['id'].' ';  
    echo '金額は'.$row['price'].'<br>';  
}
```

■商品データの検索

- ①データベースに接続

- ②SQL を生成

設定したいプレースホルダーを「?」で記述しておく

- ③prepare で実行準備

PDO クラスの prepare メソッドを使用して実行準備する

構文例 `$ps = $pdo->prepare($sql);`

④execute で実行

PDOStatement クラスの execute メソッドを使用する

SQL 文の「?」部分に設定する値を配列にして順番に渡し、実行する

構文例 `$ps->execute([値]);`

正しく実行されると TRUE、実行に失敗すると FALSE を返す

⑤実行結果を読み込む

■テーブルにデータ追加

SQL の INSERT 文でデータを追加する

「?」が複数ある場合、execute 文の引数を配列の形にして渡す

データチェック等のため、empty 関数、preg_match 関数、htmlspecialchars 関数を使用する

■テーブルデータの更新

SQL の UPDATE 文でデータを更新する

使用例

```
$pdo=new PDO('mysql:host=localhost;dbname=shop;charset=utf8','staff','password');
$sql="UPDATE product SET name=?, price=? WHERE id=?";
$ps=$pdo->prepare($sql);
if($ps->execute([htmlspecialchars($_POST['name']), $_POST['price'], $_POST['id']])) {
    echo '更新に成功しました。';
} else {
    echo '更新に失敗しました。';
}
```

■テーブルデータの削除

SQL の DELETE 文でデータを削除する

使用例

```
$pdo=new PDO('mysql:host=localhost;dbname=shop;charset=utf8','staff','password');
$sql="DELETE FROM product WHERE id=?";
$ps=$pdo->prepare($sql);
if($ps->execute([$_POST['id']])){
    echo '削除に成功しました。';
} else {
    echo '削除に失敗しました。';
}
```

■bindValue 関数

プレースホルダーに値をバインドする関数

構文 bindValue(パラメータ ID, バインドする値, PDO データ型定数)
 パラメータ ID は「?」の先頭からの位置番号（1 から順番）または「:プレースホルダー」
 データ型は、文字は「PDO::PARAM_STR」、数値は「PDO::PARAM_INT」

使用例 \$sql="UPDATE product SET name=?, price=? WHERE id=?";
 \$ps=\$pdo->prepare(\$sql);
 \$ps->bindValue(1,\$_POST['name'],PDO::PARAM_STR); //名前をバインド
 \$ps->bindValue(2,\$_POST['price'],PDO::PARAM_INT); //価格をバインド
 \$ps->bindValue(3,\$_POST['name'],PDO::PARAM_INT); //id をバインド
 \$ps->execute(); //SQL 文を実行する

■hidden パラメータ

画面上に表示されないデータを保存する場所を作る場合、type 属性を hidden にする

使用例 <input type="hidden" name="id" value="1">

■リンクにリクエストパラメータを渡す

構文 リンク先ファイル名?リクエストパラメータ名=値

使用例 削除

複数のリクエストパラメータがある場合、「&」で区切って並べる

送信されるデータは GET で送られる

7. 実用的なスクリプト

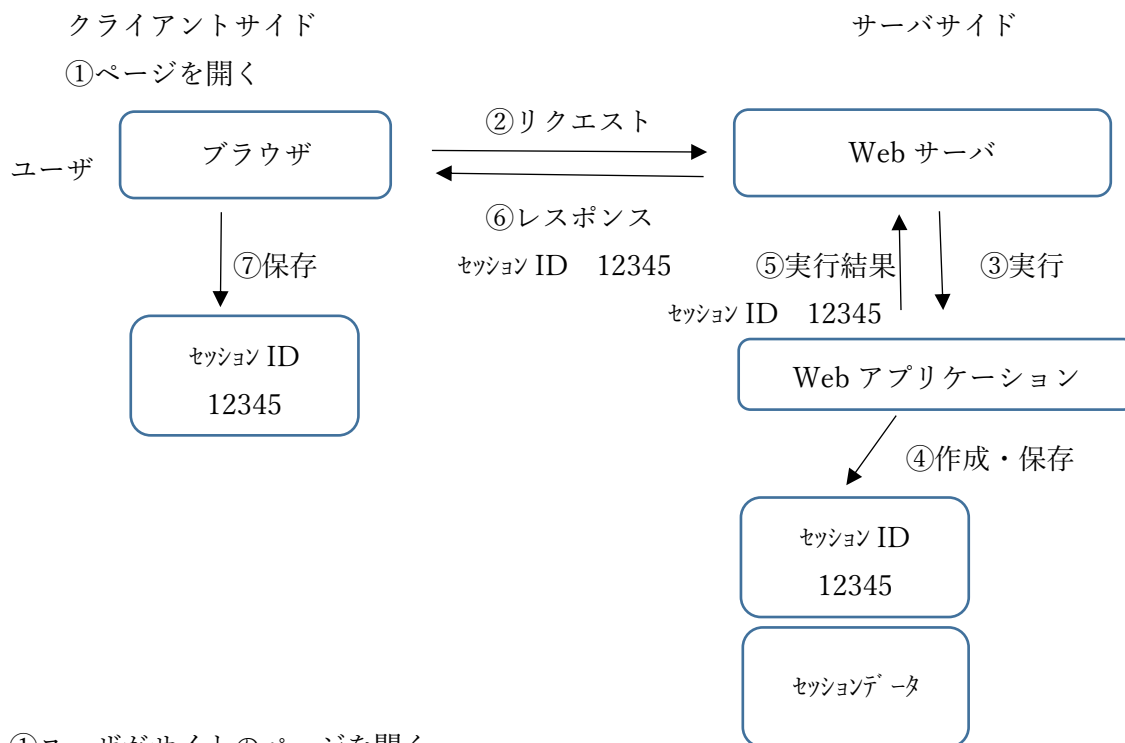
■セッション機能

各ユーザに固有のデータを格納するための仕組み

セッションを使うと、ユーザごとに異なるデータを管理することができる

ショッピングサイトでは、ログイン機能やカート機能の実現で使用する

<初回接続時>



① ユーザがサイトのページを開く

② ブラウザが Web サーバにリクエストを送信する

③ Web サーバが Web アプリケーションを実行する

④ Web アプリケーションはセッション ID とセッションデータを作成する

セッション ID は個々のセッションを区別するための識別番号

セッションデータは各セッションに属するデータで、PHP では `$_SESSION` を用いる

⑤ Web アプリケーションはセッション ID を Web サーバに渡す

⑥ Web サーバはセッション ID をレスポンスの一部としてブラウザに送信する

⑦ ブラウザは受信したセッション ID をクライアントサイドに保存する

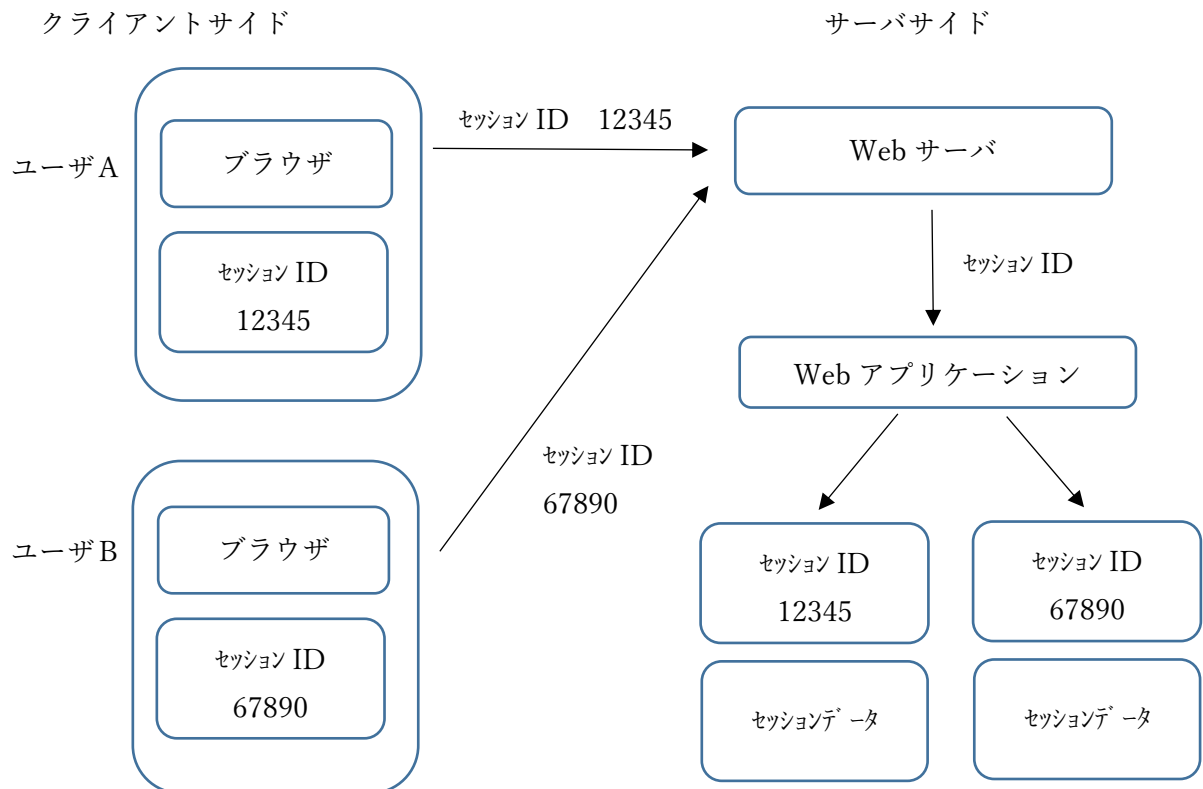
セッション ID の保存と送信には Cookie が使用される

<2 回目以降の接続時>

ブラウザはセッション ID を Cookie から取得し、Web サーバにリクエストする

Web サーバは、セッション ID を使い Web アプリを実行し、セッションデータを取得する

<複数ユーザが接続する時>



Web サーバは送信されたセッション IDをもとに、各ユーザのセッションデータを取得する

■ログイン機能

セッション機能を利用してログイン機能を実現することができる

- ①ユーザはログイン名とパスワードを入力し、Web サーバに送信
- ②Web サーバはログイン名とパスワードの組み合わせが DB に登録されているか調べる
- ③登録されていたら、セッション ID とセッションデータを作成し、ユーザに渡す
- ④ユーザはセッション ID を保存する

ユーザがログイン済かどうかは、セッションデータが取得できたかどうかで確認できる

■セッションの利用

PHP でセッションを利用するには、`session_start` 関数をスクリプトの先頭で呼び出す

構文 `<?php session_start();?>`

セッションデータは配列 `$_SESSION` を使って操作する

構文 `$_SESSION['任意の名前']=設定する値;`

使用例 `$_SESSION['customer']=1;`
`echo $_SESSION['customer'];` // 1 が表示される

構文 `$_SESSION['任意の名前']=['項目名 1'=>値 1, '項目名 2'=>値 2, ...];`

使用例 `$_SESSION['customer']=['id'=>1, 'name'=>'abc'];`
`echo $_SESSION['customer']['name'];` // abc が表示される

セッションデータに値がセットされているか調べる

構文 `isset($_SESSION['任意の名前']);`

セッションデータを削除する場合には、`unset` 関数を使用する

構文 `unset($_SESSION['任意の名前']);`

■カート機能

セッションデータに購入したい商品データを保存することで、カート機能を実現する

■セッションデータの構造

カート内の商品情報は、`$_SESSION` のなかに次のような方法で保存する

商品名 `$_SESSION['product'][商品番号]['name']`

価格 `$_SESSION['product'][商品番号]['price']`

個数 `$_SESSION['product'][商品番号]['count']`

複数の商品をカートに入れた場合には、`$_SESSION` に配列形式で保存される

■カートの初期化

ショッピングを始める時点では、カートを空の状態に初期化する

構文 `$_SESSION['任意の名前']=[];`

■DAO (Data Access Object) パターン

データベースを使用する部分（接続、検索、追加、変更、削除など）をクラス化する

機能ごとにクラス化すると、使いたいときに簡単に呼び出せる

接続先が変わったり、テーブル構成が変わった場合は、そのクラスだけ修正すればよい

■DAO クラスの利用

定義側

①DAO クラスを定義する

②DAO クラス内に、データベース接続、テーブル検索などの関数を定義する

利用側

③利用したい画面で DAO クラスを読み込む

④クラスを `new` してインスタンスを生成する

⑤処理したい関数を呼び出す

⑥取得した情報を利用する

8. Web アプリケーションの公開

■XFREE

今回は無料レンタルサーバーの XFREE を利用した

1GB まで PHP・MySQL サーバー機能が無料

以下のサイトからメールアドレスを使い申込

<https://www.xfree.ne.jp/>

XFREE

■XFREE 管理パネル

詳しいマニュアルはこちら → https://www.xfree.ne.jp/manual/man_tool_server.php



■MySQL 設定

①MySQL 追加

新しいデータベースを追加する

MySQL一覧	MySQL追加	MySQLユーザ設定	phpmyadmin
---------	----------------	------------	------------

MySQL追加

データベース名	web2bsrv_
文字コード	UTF-8

ここでは shop と入力

②MySQL ユーザ設定

新しいユーザを追加する

MySQL一覧	MySQL追加	MySQLユーザ設定	phpmyadmin
---------	---------	-------------------	------------

MySQLユーザー追加

データベースユーザ名	web2bsrv_
データベースパスワード	

ここでは
staff
password と入力

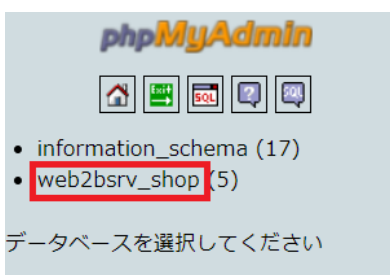
③MySQL 一覧で追加可能ユーザから権限設定ユーザを追加する

データベース名	権限設定ユーザ	追加可能ユーザ	
web2bsrv_shop	web2bsrv_masa(sv5.php.xdomain.ne.jp) 削除	web2bsrv_staff(sv5.php.xdomain.ne.jp) 追加	削除

④

phpmyadmin に接続して、テーブル作成、データ登録する

②で入力したユーザ名とパスワード
(ここでは、web2bsrv_staff と password)



左側のデータベースを選択し、
テーブル作成等を行う。

■FFFTP

Windows 専用の無料で使える FTP クライアントソフト

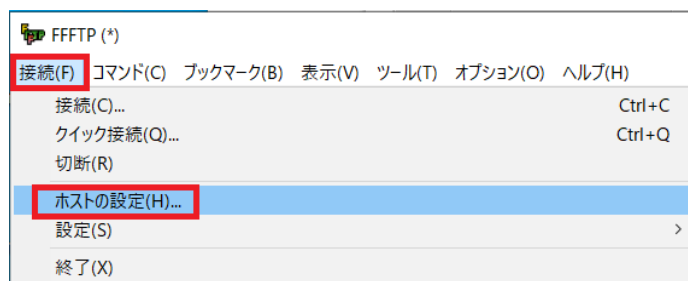
サーバへファイル転送する場合に手軽で便利

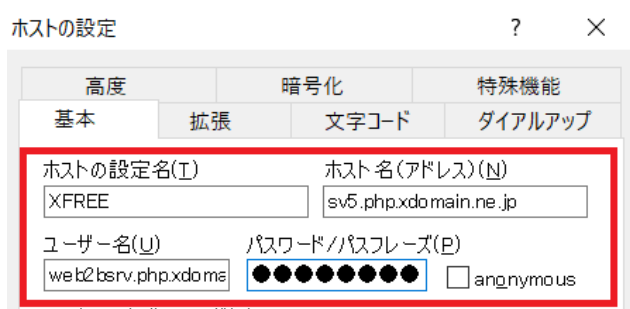
FFFTP を以下のサイトよりダウンロード・インストール

FFFTP(エフエフエフティーピー)プロジェクト日本語トップページ - OSDN

<https://osdn.jp/projects/ffftp/>

①ホストの設定

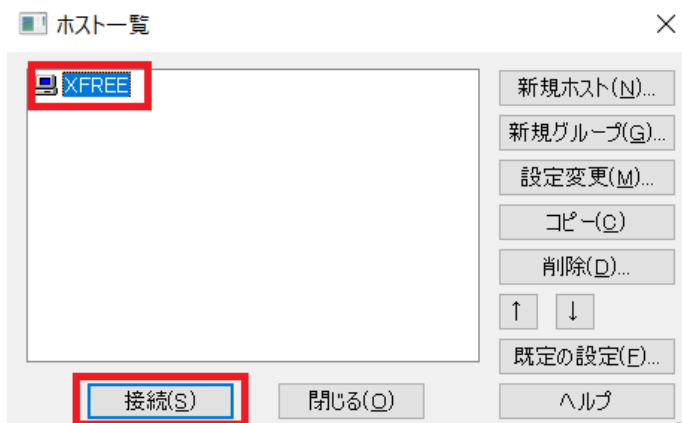
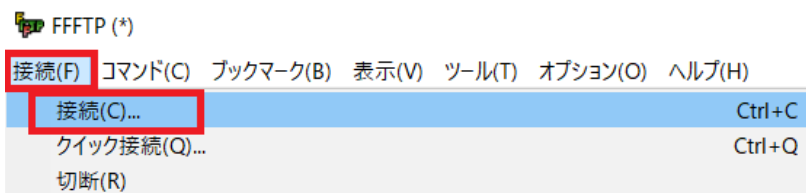




ホスト設定名は任意

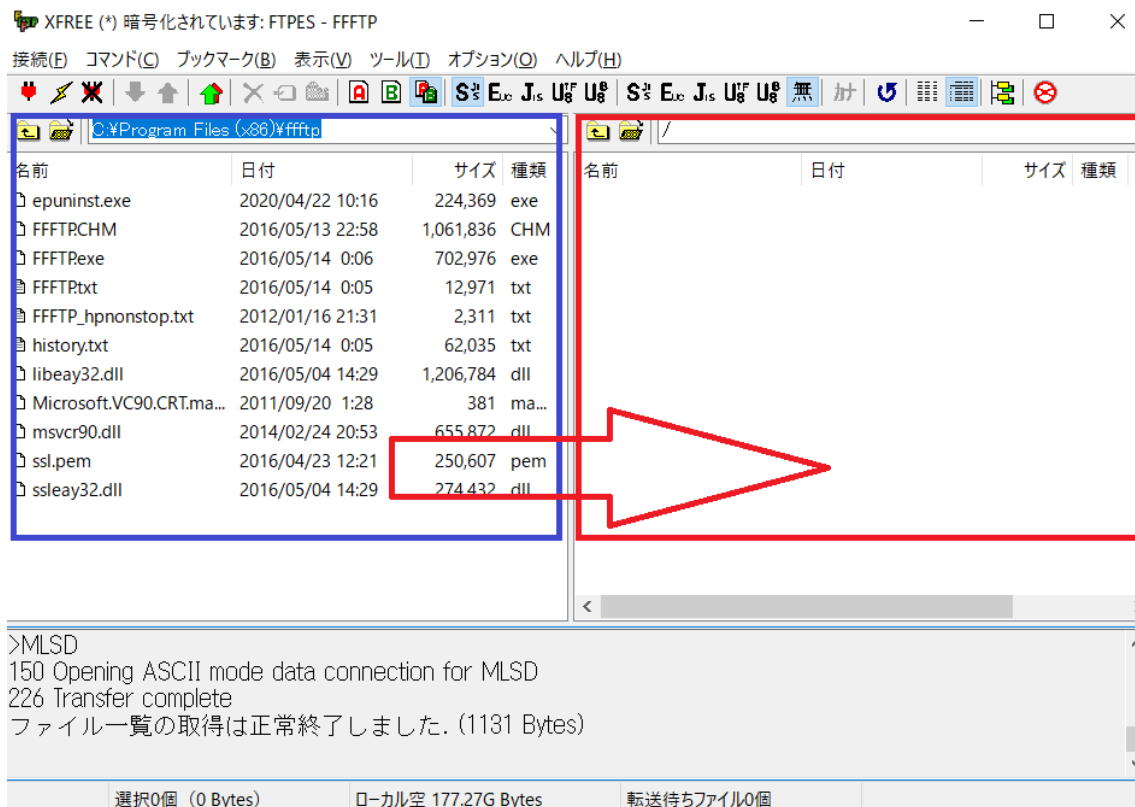
ホスト名は XFREE のサーバ情報から
ユーザ名は XFREE のドメイン設定から
パスワードは XFREE 申込時から

②ホストへ接続



③アップロード

転送元（左側）から転送先（右側）へドラッグアンドドロップでアップロード



■サイトへのアクセス

ブラウザから URL (http://ドメイン名/フォルダ名/ファイル名) を指定