DB insert & select

G's ACADEMY FUKUOKA

アジェンダ

- webの仕組み（復習）
- DBとは
- DB作成
- DB操作
- PHPからDB操作
  - 登録
  - 表示
- 課題発表→P2Pタイム

授業のルール

- 授業中は常にエディタを起動！
- 考えたことや感じたことはzoomチャットでガンガン発信！
- 質問はslackへ！ 他の人の質問にも目を通そう！（同じ質問があるかも）
- 演習時，できた人はスクショなどslackに貼ってアウトプット！
- まずは打ち間違いを疑おう！

  {} ' " ; など
- 書いたら保存しよう！（よく忘れる！）

  command + s

  ctrl + s

PHPの準備

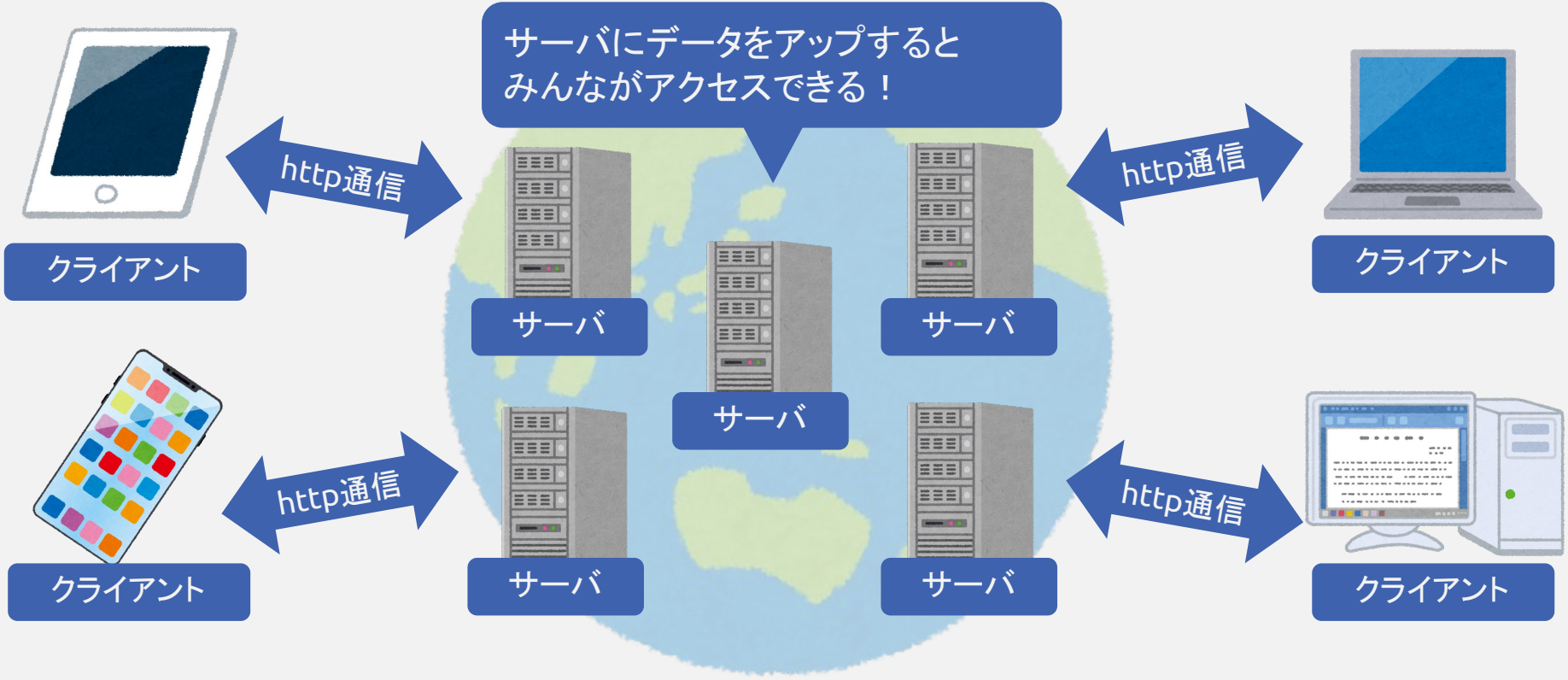
- XAMPPの起動確認
- http://localhost/のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

今日のゴール

- DBの基本を理解する！
- SQLでDBを操作する！
- PHPでDBを操作する！

webの仕組み（復習）

雑なwebの仕組み



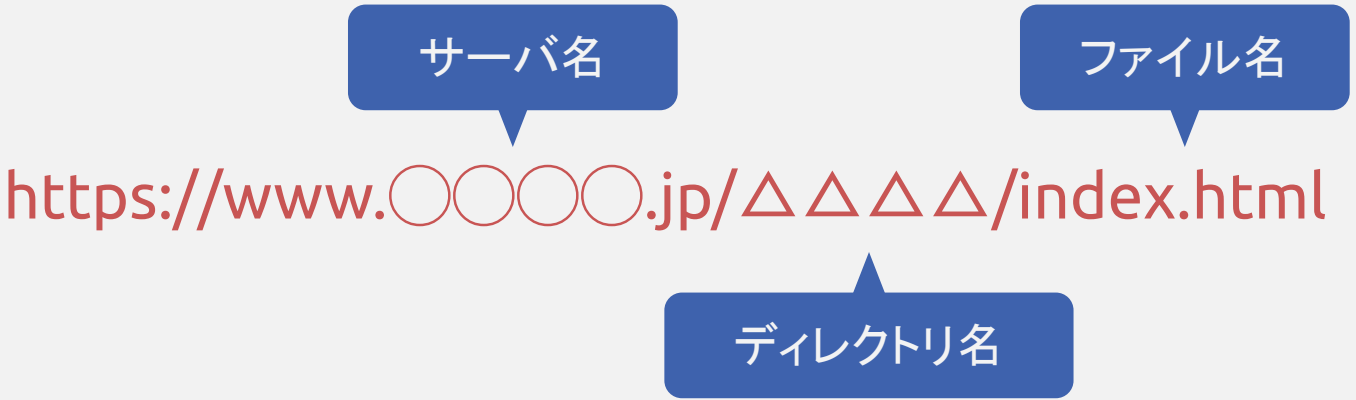
G's ACADEMY FUKUOKA

URL

■URLとは

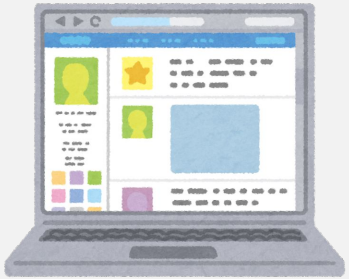
- web上にある情報（ファイル）の場所を指し示す住所.
- Uniform Resource Locatorの略（覚えなくてOK）.

■例



サーバサイド言語の仕組み

※ 言語によらず，ファイル（プログラム）はサーバ上に存在



送られてきたhtmlを実行

- こういう情報がほしい
- こういう処理をしたい
  例：index.phpにアクセス

http通信

http通信

- 処理した結果のデータ
- 構成したhtml

DBとは

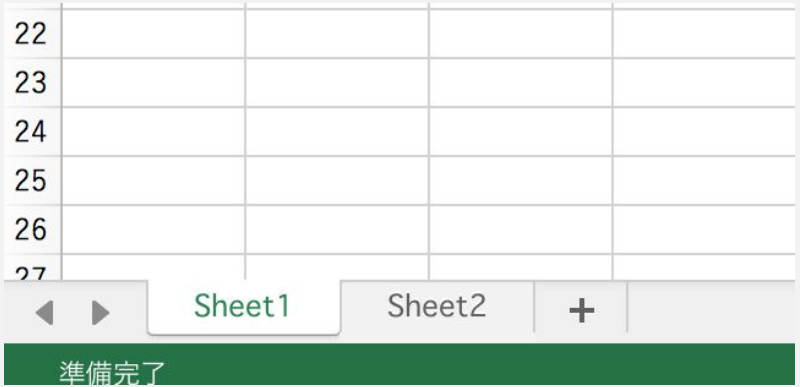
データベース（DB）とは

■DBとは

- web上にデータを保存するためのもの.
- 構造はエクセルなどと対比するとイメージしやすい！

【DB】
データベース
テーブル
レコード
カラム

【エクセル】
ファイル
シート
行
列



データベース（DB）とは

■使い所

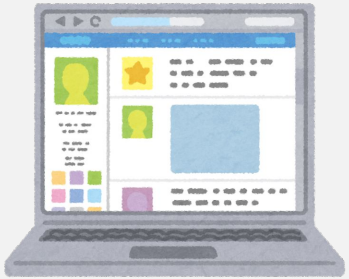
- webアプリケーションでデータを保存する場合のほとんど.

■例

- ECサイトの商品データ（商品名，画像，説明文）
- サービスに登録しているユーザの情報（ユーザ名，アドレス，etc）
- ブログの投稿内容（投稿日時，タイトル，画像，本文，etc）

データベース（DB）の動き方

サーバ上のプログラムがDBにアクセスして処理を実行！



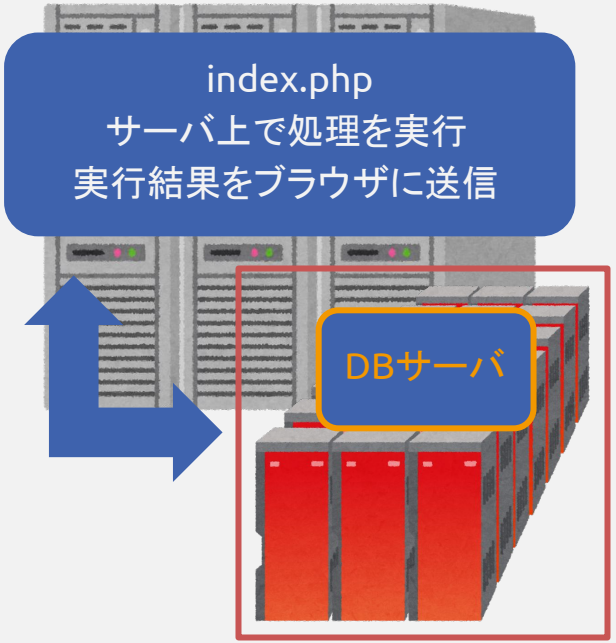
送られてきたhtmlを実行

- こういう情報がほしい
- こういう処理をしたい
  例：index.phpにアクセス

http通信

http通信

- 処理した結果のデータ
- 構成したhtml



今回はtodo管理アプリを作成！！

DB作成

DB作成

■流れ

- DBの作成
- テーブルの作成
- カラムの作成
- データの登録

DB作成

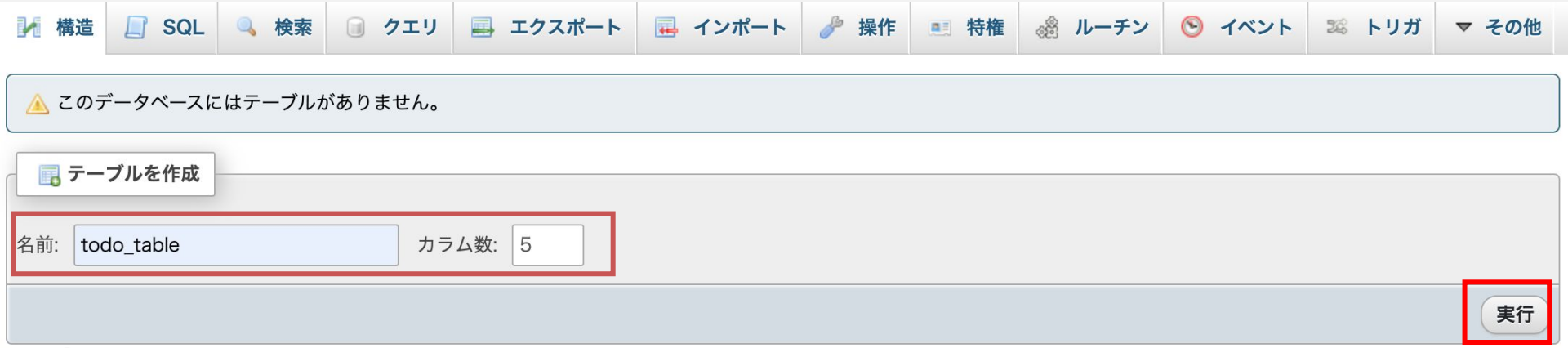
■DB準備

- http://localhost/phpmyadmin/にアクセス
- 「Databases」タブをクリック
- 「Database name」に「gsacf_クラス種別&番号2桁_受講番号2桁」入力（DB名）
- 「utf_Unicode_ci」を選択→「作成」をクリック

テーブル & カラム作成

■テーブル作成

- 左側のdb一覧から前ページで作成したDBを選択
- 名前に「todo_table」を入力（テーブル名）
- カラム数は「5」に設定
- 「Go」ボタンをクリック



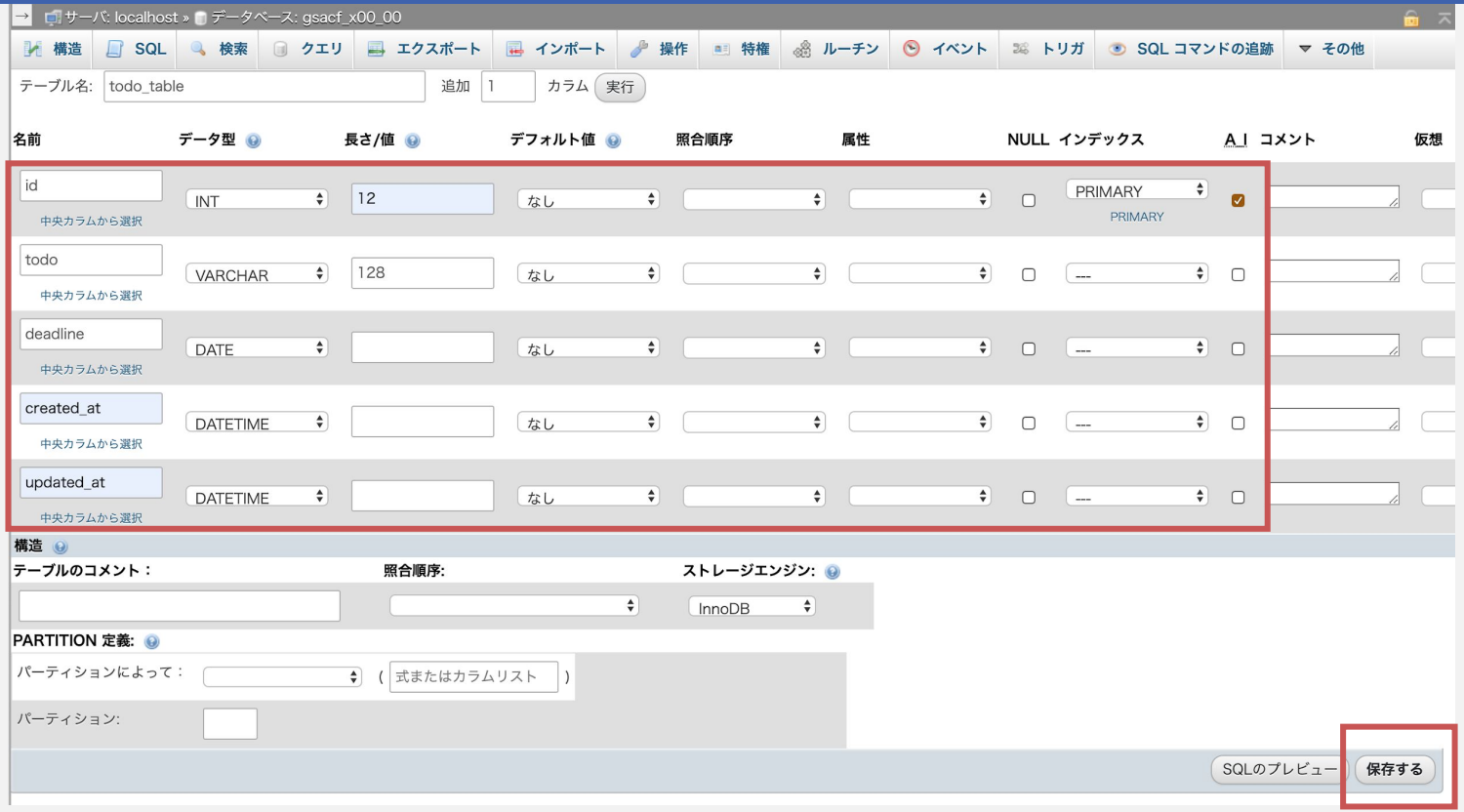
テーブル & カラム作成

■カラム作成

- 各カラムに「名前」と「データ型」を設定
- idはインデックスを「PRIMARY」に設定(重要)
- idは「AI」，commentは「NULL」にチェック(重要)
- 左下の「保存する」をクリック(超重要)

- 次ページの内容を見ながら同じように入力！！

テーブル & カラム作成


サーバ: localhost » データベース: gsacf_x00_00
構造
SQL
検索
クエリ
エクスポート
インポート
操作
特権
ルーチン
イベント
トリガ
SQL コマンドの追跡
その他
テーブル名: todo_table
追加 1 カラム 実行
名前
データ型
長さ/値
デフォルト値
照合順序
属性
NULL
インデックス
A_I
コメント
仮想
id
INT
12
なし
PRIMARY
todo
VARCHAR
128
なし
deadline
DATE
なし
created_at
DATETIME
なし
updated_at
DATETIME
なし
中央カラムから選択
構造
テーブルのコメント：
照合順序:
ストレージエンジン:
InnoDB
PARTITION 定義:
パーティションによって：
式またはカラムリスト
パーティション:
SQLのプレビュー
保存する

DB操作

DBの操作

■SQL

- DBの操作には「SQL」を使います.
- PHPでDBを操作するときは, コード内でSQL文を実行します.
- フレームワークなどではコード内では実行しない場合もあります.

DBの操作

■基本のSQL（まずはこれを押さえましょう！）

- INSERT：　データの「登録」
- SELECT：　データの「表示」
- UPDATE：　データの「更新」
- DELETE：　データの「削除」

※SQL文は大文字で記載していますが，小文字でも動作します．

　他の言語と組み合わせる際に区別しやすいよう大文字で記載．

SQL構文：INSERT (データ登録)

```
// INSERT文の基本構造
INSERT INTO テーブル名(カラム1, カラム2, ...)VALUES(値1, 値2, ...);

// 例
INSERT INTO gs_table (id, name, email, detail, created_at)
VALUES(NULL, 'gs_00', 'gsf00@gs.com', 'test', sysdate());

// ↑スペースの都合上2行にしているが1行にまとめて記述する！
// カラム名の数と値の数が一致するよう注意！！
```

SQL構文：SELECT (データ取得)

```
// SELECT文の基本構造
SELECT 表示するカラム名 FROM テーブル名;

// 例
SELECT * FROM gs_table;                          -- 「*」で全て指定
SELECT name FROM gs_table;                    -- 1つのカラムを指定
SELECT name, email FROM gs_table;                -- 複数カラム指定
SELECT * FROM gs_table WHERE name='gs_00';

// ※「WHERE」を使用して値の条件を指定できる
```

SQL構文：SELECT文のオプション

```
// 演算子の使用
SELECT * FROM gs_table WHERE id = 1;       -- 「==」ではない！
SELECT * FROM gs_table WHERE id >= 1;
SELECT * FROM gs_table WHERE id >= 1 AND id <= 3;

// あいまい検索
SELECT * FROM gs_table WHERE email LIKE 'gs%';
SELECT * FROM gs_table WHERE email LIKE '%gmail.com';
SELECT * FROM gs_table WHERE email LIKE '%@%';
```

SQL構文：SELECT文のオプション

```
// ソート
SELECT * FROM gs_table ORDER BY id DESC;
SELECT * FROM gs_table ORDER BY name, email ASC;
// ※DESC→降順，ASC→昇順

// 表示件数の制限
SELECT * FROM gs_table LIMIT 5;        -- 5件のみ表示

// 上記の組み合わせ
SELECT * FROM gs_table ORDER BY id DESC LIMIT 5;
```

SQL構文：UPDATE（データ更新）　←次回

```
// UPDATE文の基本構造
UPDATE テーブル名 SET 変更データ WHERE 選択データ;

// 例
UPDATE gs_table SET name='gs99' WHERE id = 1;
// 必ずWHEREを使用！！→忘れると全てのデータが更新されます．．！
```

SQL構文：DELETE（データ削除）　←次回

```
// DELETE文の基本構造
DELETE FROM テーブル名;

// 例
DELETE FROM gs_table;                    -- 全消去
DELETE FROM gs_table WHERE id = 2;       -- 指定データのみ
// WHEREで指定しないとテーブルのデータが全滅する！！
// DELETEすると復旧できないので注意！！
```

【参考】SQL練習

https://sqlzoo.net/

- 初歩から応用までSQL問題が出題.
- 「0」「1」くらいまでで当面OK！

PHPからDB操作

データ登録処理

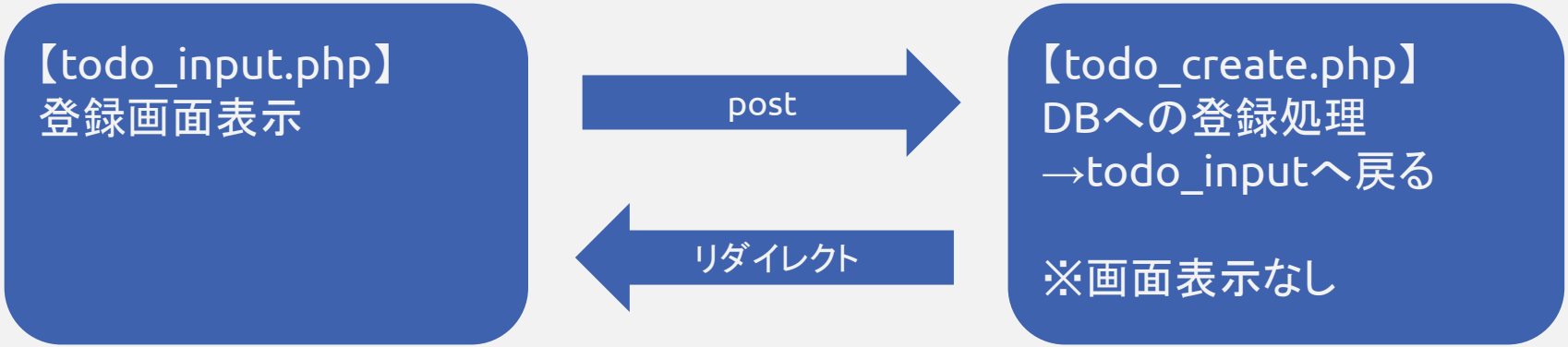
PHPでのDB操作の流れ（登録処理）

■処理の流れ

①todo_input.phpで入力されたデータをtodo_create.phpへ送信（post）

②todo_create.phpでデータを受け取り，DBへの登録処理を実行

③登録完了後，todo_input.phpへ移動．todo_create.phpでは画面表示なし）



form準備

todo_input.php

```
<form action="todo_create.php" method="POST">
  <fieldset>
    ...
    <div>
      todo: <input type="text" name="todo">
    </div>
    <div>
      deadline: <input type="date" name="deadline">
    </div>
    <div>
      <button>submit</button>
    </div>
  </fieldset>
</form>
```

name属性を指定しないと
phpがデータを受け取れない！

データ受け取りの確認

todo_create.php

```
// データ受け取りのときにまずやること
var_dump($_POST);
exit();

// 解説
// POSTで送信された値は$_POSTで受け取る
//  （前回と同じ）
```

データ受け取り

todo_create.php

```
// 入力チェック（未入力の場合は弾く，commentのみ任意）
if (
  !isset($_POST['task']) || $_POST['task']=='' ||
  !isset($_POST['deadline']) || $_POST['deadline']==''
) {
  exit('ParamError');
}

// 解説
// 「ParamError」が表示されたら，必須データが送られていないことがわかる
```

必須項目が送信されていない場合はエラーにする！

【参考】エラーを出す意味

■どこで失敗したのかをわかるようにする！

- PHPではエラーを見つけづらい．．．
- どこでエラーが出ているのかわからないと詰む．
- エラーにも種類がある！

=> どこでうまくいっていないのかを把握できるようにエラーの処理を記述！

データ受け取り

todo_create.php

```
// データを変数に格納
$todo = $_POST['todo'];
$deadline = $_POST['deadline'];
```

$_POST['name属性の値']で受け取れる.
変数に入れよう！
（getの場合は$_GET）

DB接続

todo_create.php

```
// 「dbname」「port」「host」「username」「password」を設定
$dbn ='mysql:dbname=YOUR_DB_NAME;charset=utf8;port=3306;host=localhost';
$user = 'root';
$pwd = '';// （空文字）

try {
  $pdo = new PDO($dbn, $user, $pwd);
} catch (PDOException $e) {
  exit('dbError:'.$e->getMessage());
}

// 「dbError:...」が表示されたらdb接続でエラーが発生していることがわかる.
```

データ登録SQL作成→実行

todo_create.php

```
// SQL作成&実行
$sql = 'INSERT INTO todo_table(id, todo, deadline, created_at, updated_at)
          VALUES(NULL, :todo, :deadline, sysdate(), sysdate())';
```

変数をバインド変数（:todo）に格納！！

```
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':todo', $todo, PDO::PARAM_STR);
$stmt->bindValue(':deadline', $deadline, PDO::PARAM_STR);
$status = $stmt->execute();   // SQLを実行
```

【補足】バインド変数

■SQLインジェクション（ハッキング手法の一つ）

下記のようにコードを記述した場合．．．

```
$query = "SELECT * FROM user WHERE id = '$user_id'";
```

$user_idに「' or 'A' = 'A」を入れると，下記と同じ意味になってしまう！

```
SELECT * FROM user;
```

=> 不正にデータを取得できてしまう！！！

バインド変数を用いることで，SQL文として実行されないようにする！

=> 安心！！

データ登録実行後の処理

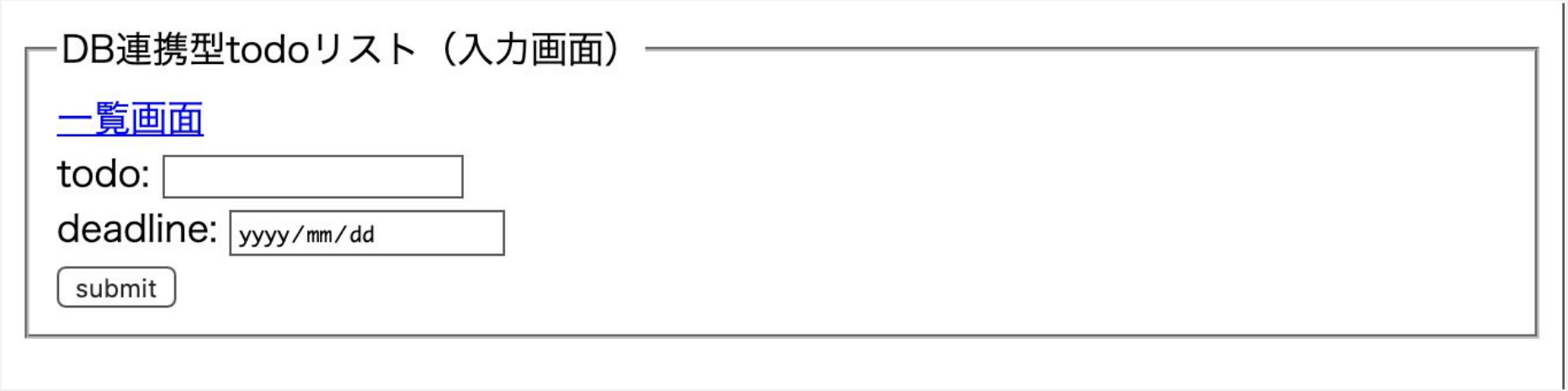
todo_create.php

```
// 失敗時にエラーを出力し，成功時は登録画面に戻る
if ($status==false) {
  $error = $stmt->errorInfo();
  // データ登録失敗次にエラーを表示
  exit('sqlError:'.$error[2]);
} else {
  // 登録ページへ移動
  header('Location: todo_input.php');
}
```

todo_input.phpに移動！

動作確認（登録画面）

todo_input.php



動作確認（登録処理）

phpmyadminでデータが入っていればOK！

データ表示処理

PHPでのDB操作の流れ（表示処理）

■処理の流れ

①表示ファイル（todo_read.php）へアクセス時，DB接続

②データ抽出用SQL作成→実行

③取得したデータを埋め込んで画面を表示

※必要に応じて，並び替えやフィルタリングを実施する．

DB接続（insert.phpと同じ）

todo_read.php

```
// 「dbname」「port」「host」「username」「password」を設定
$dbn ='mysql:dbname=YOUR_DB_NAME;charset=utf8;port=3306;host=localhost';
$user = 'root';
$pwd = '';// （空文字）

try {
  $pdo = new PDO($dbn, $user, $pwd);
} catch (PDOException $e) {
  exit('dbError:'.$e->getMessage());
}

// 「dbError:...」が表示されたらdb接続でエラーが発生していることがわかる.
```

データ取得SQL作成

todo_read.php

```
// SELECT文！
$sql = 'SELECT * FROM todo_table';
$stmt = $pdo->prepare($sql);
$status = $stmt->execute();
```

実行を忘れずに！！

データ表示処理

todo_read.php

```
$view='';
if ($status==false) {
  $error = $stmt->errorInfo();
  exit('sqlError:'.$error[2]);
} else {
  $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
  $output = "";
  foreach ($result as $record) {
    $output .= "<tr>";
    $output .= "<td>{$record["deadline"]}</td>";
    $output .= "<td>{$record["todo"]}</td>";
    $output .= "</tr>";
  }
}
```

失敗時はエラーを出す！

fetchAll()で全部取れる！
あとは配列の処理！！

データ表示処理

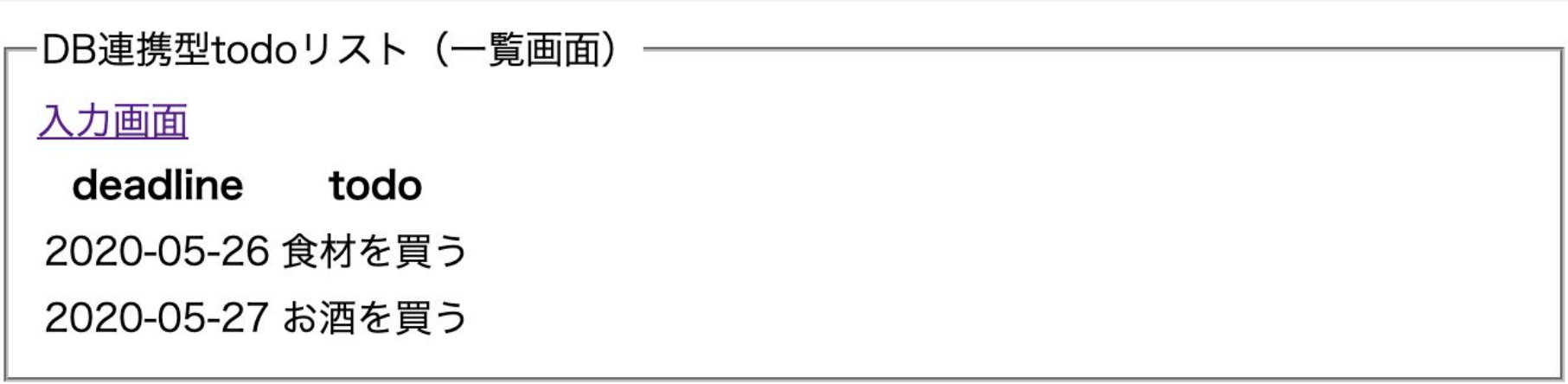
todo_read.php

```
// html部分にデータを追加

<tbody>
  <!-- ここに<tr><td>deadline</td><td>todo</td><tr>の形でデータが入る -->
  <?= $output ?>
</tbody>
```

動作確認（表示画面）

todo_read.php



課題

DB連携アプリ（登録処理 / 表示処理）

■DBを使用したアプリを実装しよう！

DB名：　　今回作成したものを使用

テーブル名：　　自由に！（todo_tableの構成は変更しないこと！）

■基本は下記ファイル（処理）を作成！（ファイル構成は変更してもOK！）

- index.php　（登録画面）
- insert.php　（登録処理）
- select.php　（表示処理）

■卒制のプロトタイプ的な作品とか，SNS的なものとか！

提出は次回授業前まで！！

P2Pタイム

まずはチーム内で解決を目指す！
訊かれた人は苦し紛れでも応える！！