

DB insert & select

Contents

- 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

rules...

- 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slackのガヤチャンネルでガンガン発信！
- 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

以下3点ができているか確認しよう！

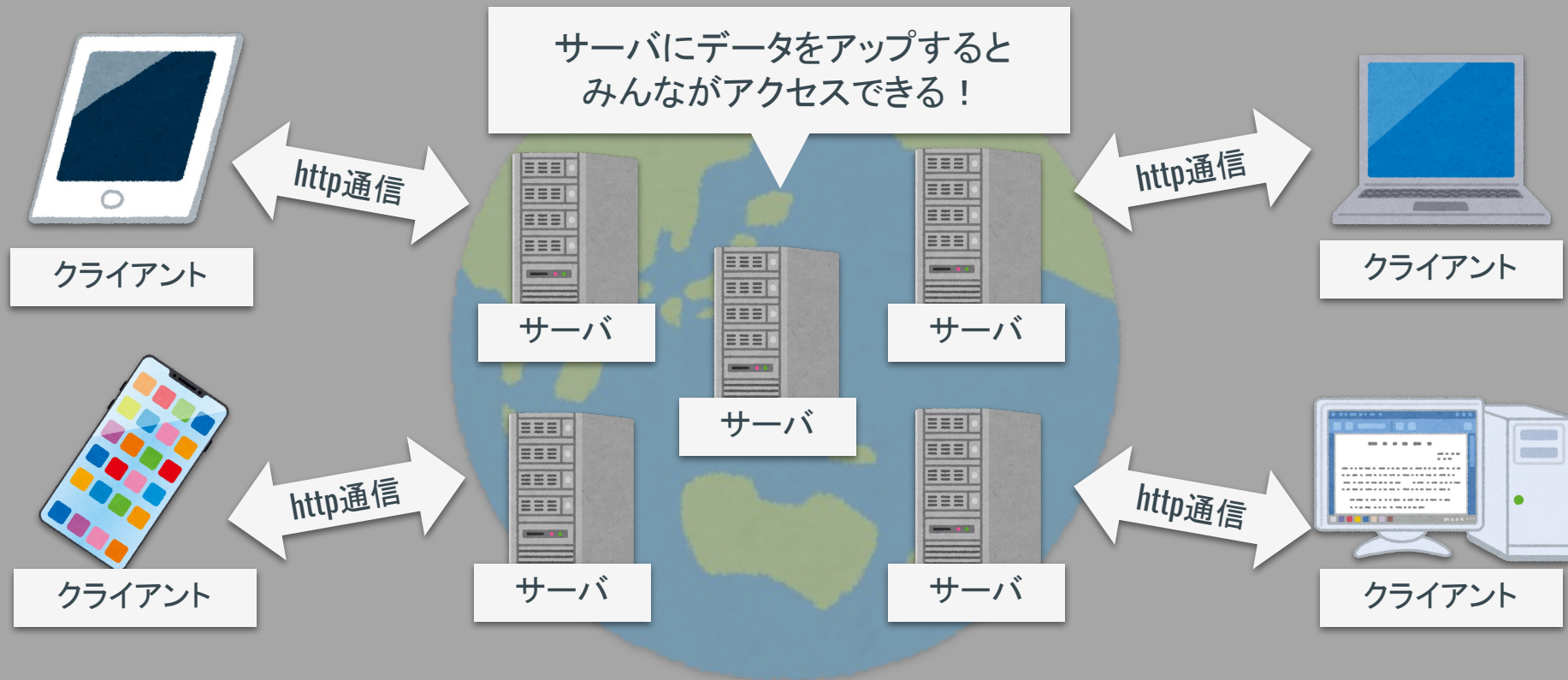
- XAMPPの起動確認
- <http://localhost/>のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

Goal

- 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

雑なwebの仕組み



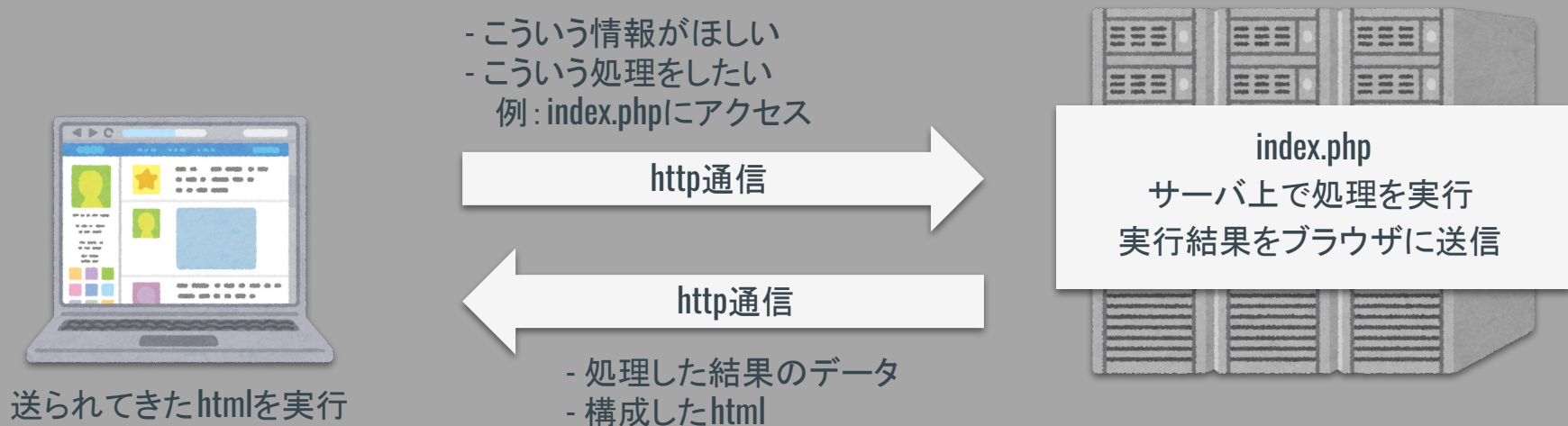
URL

- 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



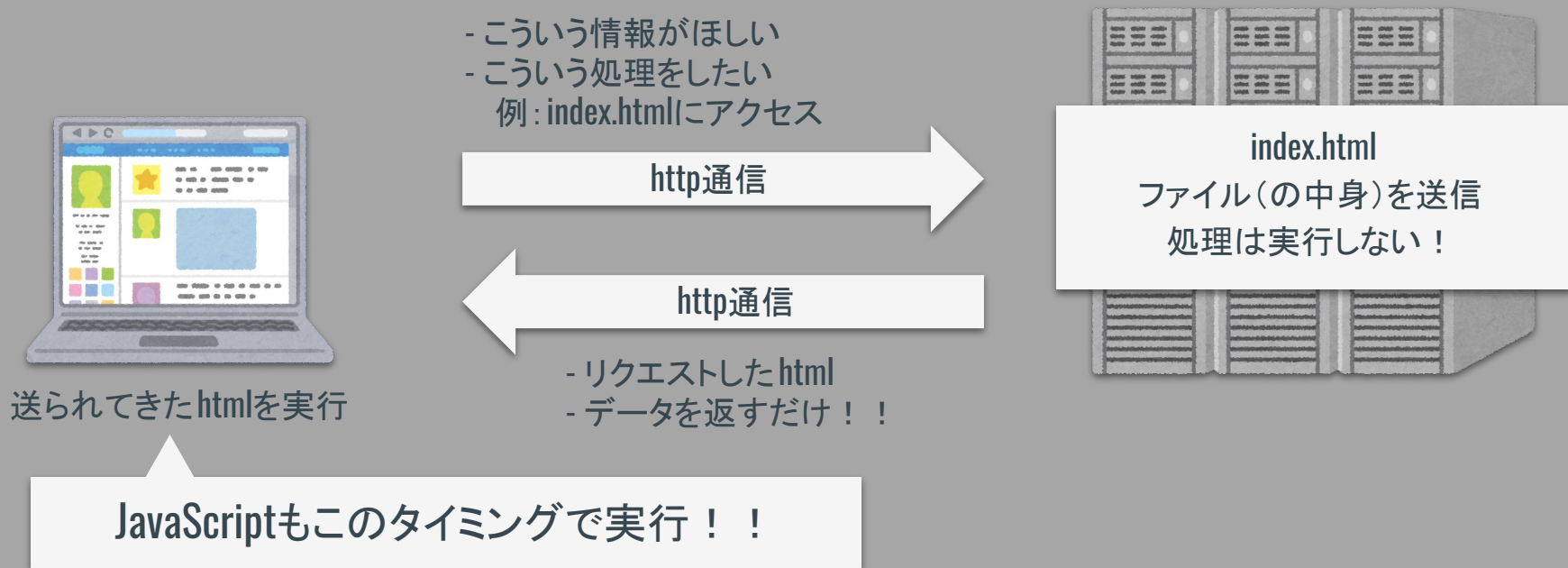
サーバサイド言語の動き方

※ 言語によらず, ファイル(プログラム)はサーバ上に存在



クライアントサイド言語の動き方

※ 言語によらず、ファイル(プログラム)はサーバ上に存在



DB(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

22				
23				
24				
25				
26				
27				

◀ ▶

Sheet1

Sheet2

+

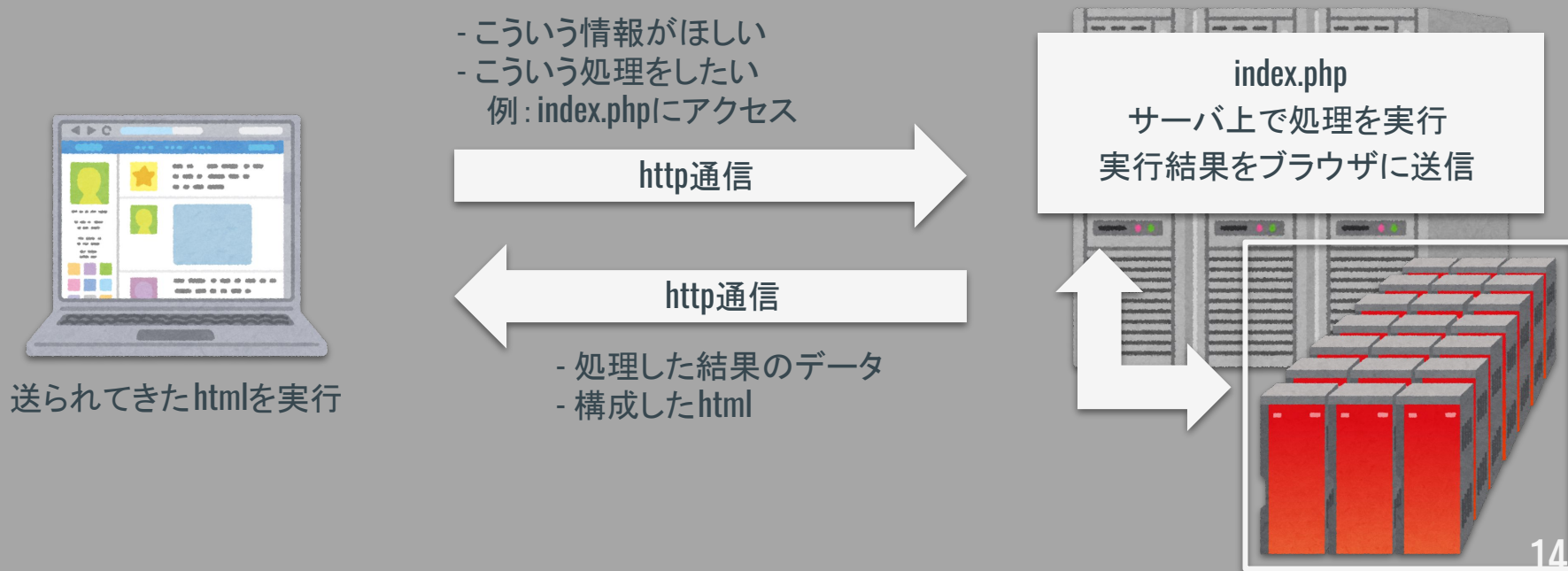
準備完了

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)

DBの動き方

サーバ上のプログラム(PHPなど)がDBにアクセスして処理を実行！



DBの作成

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」を入力(DB名)
- 「utf_Unicode_ci」を選択→「作成」をクリック

The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of databases including 'gs', 'gsf', 'information_schema', 'mydb', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. The main panel is titled 'データベース' (Databases) and contains a section 'データベースを作成する' (Create database). In this section, the 'データベース名' (Database name) field is highlighted with a red box and contains the text 'gsacf_d06_05'. To its right is a dropdown menu showing 'utf8_unicode_ci'. A callout bubble points to the database name field with the text '例 : gsacf_d06_05'. Below the name field is a 'フィルタ' (Filter) section with a label '含まれている文字:' (Characters included:).

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

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	その他
----	-----	----	-----	--------	-------	----	----	------	------	-----	-----

⚠ このデータベースにはテーブルがありません。

テーブルを作成

名前: カラム数:

実行

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」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	コメント	仮想
id <small>中央カラムから選択</small>	INT	12	なし			<input type="checkbox"/>	PRIMARY PRIMARY	<input checked="" type="checkbox"/>	
todo <small>中央カラムから選択</small>	VARCHAR	128	なし			<input type="checkbox"/>	---	<input type="checkbox"/>	
deadline <small>中央カラムから選択</small>	DATE		なし			<input type="checkbox"/>	---	<input type="checkbox"/>	
created_at <small>中央カラムから選択</small>	DATETIME		なし			<input type="checkbox"/>	---	<input type="checkbox"/>	
updated_at <small>中央カラムから選択</small>	DATETIME		なし			<input type="checkbox"/>	---	<input type="checkbox"/>	

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「作成」
- SELECT データの「参照」
- 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 (データ作成)

-- 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に注意！！

SQL構文:SELECT (データ参照)

-- 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;

SELECT name FROM gs_table;

SELECT name, email FROM gs_table;

SELECT * FROM gs_table WHERE name='gs_00';

-- 「*」で全て指定

-- 1つのカラムを指定

-- 複数カラム指定

-- ※「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)

-- 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;

-- **【重要】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)

```
// 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からデータ操作

エラー表示用の設定

エラー表示用の設定

PHPではデフォルトでエラーが表示されない設定となっている。

開発時，エラーメッセージを確認しやすいよう設定の上書きを行う。

1. プロジェクトのフォルダ内に`.htaccess`ファイルを作成する。
2. `.htaccess`ファイルに下記の内容を追記して保存する。

```
php_flag display_errors On
```

これで，PHPのコードが実行されてエラーが発生した場合に画面で確認することができる。

データ作成処理(Create)

Create処理の流れ

処理の流れ

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

【todo_input.php】
登録画面表示

post

【todo_create.php】
DBへの登録処理
→todo_inputへ戻る

リダイレクト

※画面表示なし

送信側の処理(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！**

データ受信側の処理(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）
```

データ受信側の処理(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['todo']) || $_POST['todo']=='' ||
    !isset($_POST['deadline']) || $_POST['deadline']==''
) {
    exit('ParamError');
}
```

必須項目が送信されていない
場合はエラーにする！

```
// 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
```

【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...
- データを扱うので、異常なデータなどが作成されるとまずい.
- 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！

データ受信側の処理(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 $e) {
    echo json_encode(["db error" => "{$e->getMessage()}"]);
    exit();
}
// 「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。
```

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)

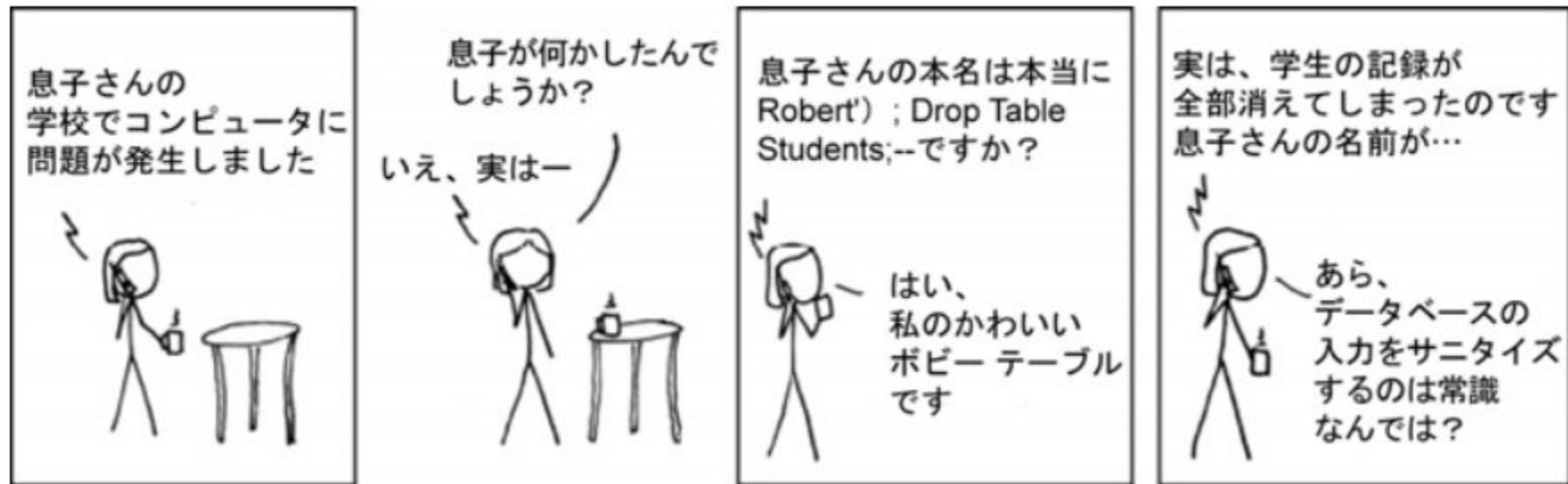
- 下記のようにコードを記述した場合...
- `$query = "SELECT * FROM user WHERE id = '$user_id'";`
- `$user_id`に「' or 'A' = 'A」を入れると、下記と同じ意味になってしまう！
- `SELECT * FROM user;`

-> 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！！

【参考】SQLインジェクションによるデータ消去



<https://www.oracle.com/technetwork/ip/database/features/plsql/how-to-write-injection-proof-plsql-132808-ja.pdf>

データ作成実行後の処理(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(入力画面)

DB連携型todoリスト（入力画面）

[一覧画面](#)

todo:

deadline:

動作確認(データ作成処理)

表示 構造 SQL 検索 挿入 エクスポート インポート 特権 操作

✓ 行 0 - 0 の表示 (合計 1, クエリの実行時間: 0.0010 秒。)

```
SELECT * FROM `todo_table`
```

☐ プロファイリング [インラインを編集する] [編集]

☐ すべて表示 | 行数: 25 | 行フィルタ: このテーブルを検索

+ オプション

	id	todo	deadline	created_at	updated_at
<input type="checkbox"/> 編集 <input type="checkbox"/> コピー <input type="checkbox"/> 削除	1	食材を買う	2020-05-26	2020-05-26 11:59:07	2020-05-26 11:59:07

☐ すべてチェックする | チェックしたものを: ☐ 編集 ☐ コピー ☐ 削除 ☐ エクスポート

phpmyadminでデータが入っていればOK!

練習

DBにデータを追加する処理を実装しよう！

- todo_input.phpにフォームを作成
- todo_create.phpで
 - データを受け取る
 - DBに接続
 - SQL文を書いて実行
- phpmyadminでデータが入っていることを確認！

データ参照(&表示)処理 (read)

PHPでのDB操作の流れ(参照処理)

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. データ参照用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(todo_create.phpと同じ)

```
// 「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 $e) {
    echo json_encode(["db error" => "{$e->getMessage()}"]);
    exit();
}
// 「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。
```

データ参照SQL作成

// 参照はSELECT文！

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

実行を忘れずに！

// \$statusにSQLの実行結果が入る（取得したデータではない点に注意）

データを表示しやすいようにまとめる

```
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>";  
    }  
}
```

失敗時はエラー出力

fetchAll()で全部取れる！
あとは配列の処理！！

HTML部分に作成したデータを埋め込み

```
// html部分にデータを追加
```

```
<tbody>
```

```
  <!-- ↓に<tr><td>deadline</td><td>todo</td><tr>の形でデータが入る -->
```

```
  <?= $output ?>
```

```
</tbody>
```

動作確認(データ参照&画面表示処理)

DB連携型todoリスト (一覧画面)

入力画面

deadline	todo
----------	------

2020-05-26	食材を買う
------------	-------

2020-05-27	お酒を買う
------------	-------

練習

DBのデータを読み出して表示する処理を実装しよう！

- todo_read.phpで
 - DBに接続
 - SQL文を書いて実行
 - 取得したデータをHTMLに埋め込み

課題

卒業制作プロトタイプ実装開始！！

プロトタイプを実装しよう！

- DB名： 卒業制作用のものを準備しよう！
- 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！（ファイル構成や名前は変更してOK！）

- データ入力画面のphpファイル
- データ作成処理のphpファイル
- 参照&表示処理のphpファイル

まずは「何をDBに保存するか」「何を画面に表示するか」を考えよう！！

DB使用時の課題提出についての注意

DB使用时にはDBに保存されたデータも必要.

- DBのデータ(テーブルなど)もまとめて提出！
- 詳しい方法は次ページ以降で解説

注意点

- 「DB名」をReadmeファイルに明記すること！！

phpmyadminにアクセス

phpMyAdmin

サーバ: localhost

データベース SQL 状態 ユーザアカウント エクスポート インポート 設定 レプリケーション その他

最近 お気に入り

新規作成

- gs
- gsacfd04_00**
- gsf
- information_schema
- mydb
- mysql
- new_db
- performance_schema
- phpmyadmin
- test
- web_bc_db00

一般設定

自分の「DB名」をクリック！
※テーブル名ではない！！

言語 - Language 日本語 - Japanese

テーマ: pmahomme

フォントサイズ: 82%

詳細設定

データベースサーバ

- サーバ: Localhost via UNIX socket
- サーバの種類: MariaDB
- サーバの接続: SSLは使用されていません
- サーバのバージョン: 10.3.16-MariaDB - Source distribution
- プロトコルバージョン: 10
- ユーザ: root@localhost
- サーバの文字セット: cp1252 West European (latin1)

ウェブサーバ

- Apache/2.4.39 (Unix) OpenSSL/1.0.2s PHP/7.3.6 mod_perl/2.0.8-dev Perl/v5.16.3
- データベースクライアントのバージョン: libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: 7cc7cc96e675f6d72e5cf0f267f48e167c2abb23 \$
- PHP 拡張: mysqli curl mbstring
- PHP のバージョン: 7.3.6

データのエクスポート

The screenshot shows the phpMyAdmin web interface. The top navigation bar includes buttons for '構造' (Structure), 'SQL', '検索' (Search), 'クエリ' (Query), 'エクスポート' (Export), 'インポート' (Import), '操作' (Operations), '特権' (Privileges), 'ルーチン' (Routines), 'イベント' (Events), and 'その他' (Others). The 'エクスポート' button is highlighted with a red rectangular box. A dark blue callout box with a white arrow points to this button, containing the text 「エクスポート」をクリック. The left sidebar shows a tree view of databases, with 'gsacfd04_00' selected. Below the database list, there are sections for 'フィルタ' (Filter) and 'テーブル' (Tables). The 'テーブル' section shows a table named 'php02_table' with 11 rows, InnoDB engine, utf8_unicode_ci collation, and a size of 16 KiB. At the bottom, there is a section for 'テーブルを作成' (Create Table) with input fields for '名前' (Name) and 'カラム数' (Number of Columns) set to 4, and an '実行' (Execute) button.

phpMyAdmin

最近 お気に入り

新規作成

gs

gsacfd04_00

新規作成

php02_table

gsf

information_schema

mydb

mysql

new_db

performance_schema

phpmyadmin

test

web_bc_db00

サーバ: localhost » データベース: gsacfd04_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 その他

フィルタ

含まれている文字:

テーブル 操作

☐ php02_table ☆ 表示 構造 検索

1 テーブル 合計

11 InnoDB utf8_unicode_ci 16 KiB 0 バイト

↑ ☐ すべてチェックする チェックしたものを:

印刷 データ辞書

テーブルを作成

名前: カラム数: 4

実行

sqlファイルを保存して課題フォルダに追加！

mydb
mysql
new_db
performance_schema
phpmyadmin
test
web_bc_db00

エクスポート方法:

☒ 簡易 - 最小限のオプションだけ表示
☐ 詳細 - 可能なオプションをすべて表示

フォーマット:

SQL

エンコーディングへの変換:

☒ なし ☐ EUC ☐ SJIS
☐ 全角カナに変換する

実行

特に変更せず実行をクリック！

ダイアログが表示されるので、任意の場所にsqlファイルを保存！
このsqlファイルを課題フォルダに入れて一緒に提出！

締切は厳守！！

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！！