RDB, GROUP BY & JOIN

Contents

- webの仕組み（復習）
- RDBの構造
- テーブルの追加
  - ユーザテーブル
  - いいねテーブル
- いいねボタンの追加とdb操作
- いいね数の表示
- 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！

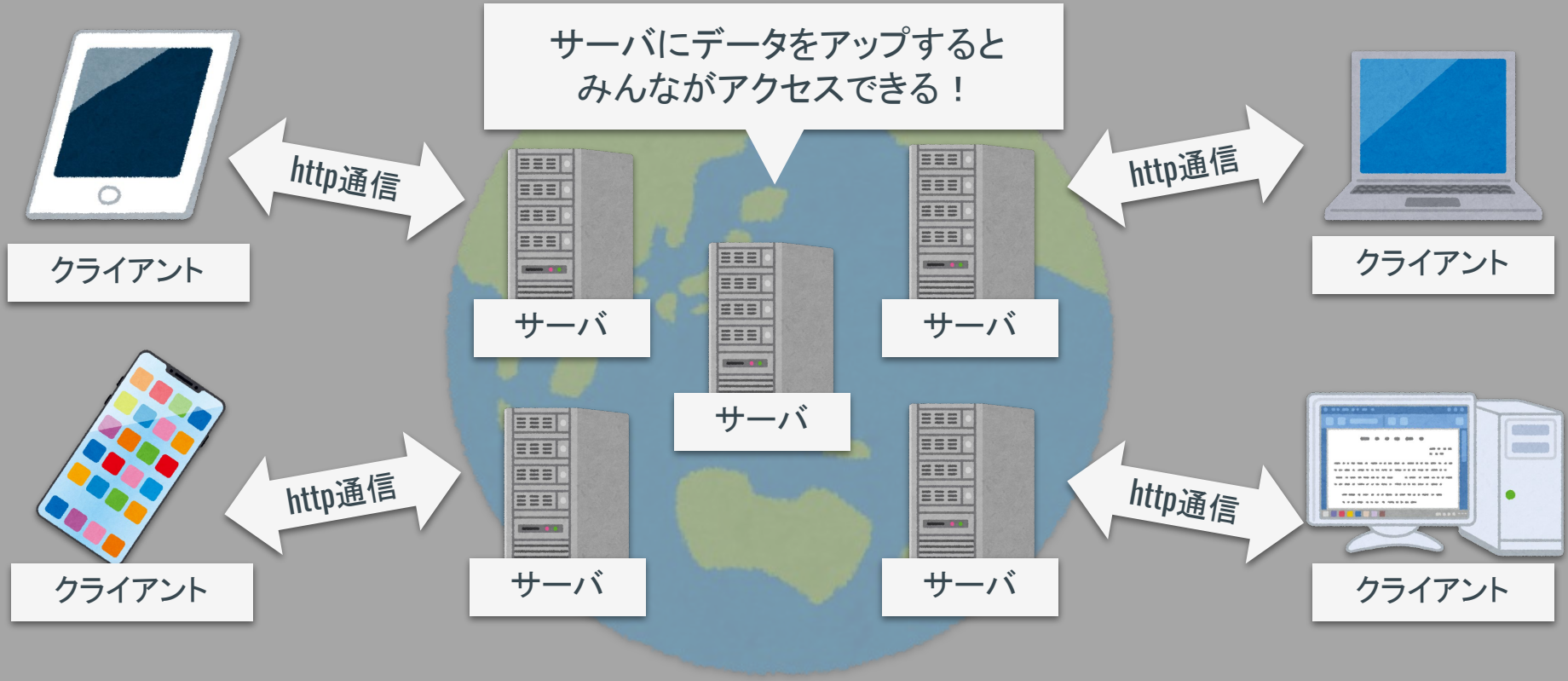
- XAMPPの起動確認
- http://localhost/のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

Goal

- RDB構造を把握！
- RDBの考え方を知る！
- 複数テーブルを操作！

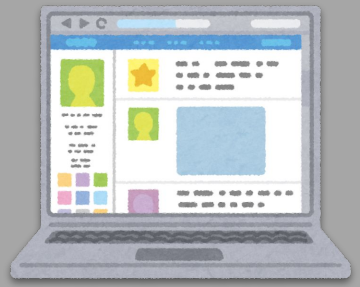
webの仕組み

雑なwebの仕組み


サーバにデータをアップすると
みんながアクセスできる！
http通信
http通信
http通信
http通信
クライアント
クライアント
クライアント
クライアント
サーバ
サーバ
サーバ
サーバ
サーバ

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



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

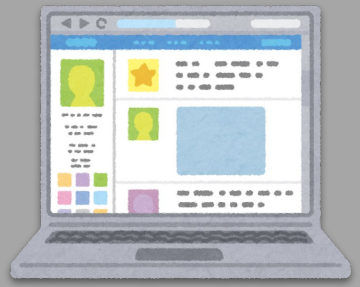
http通信

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！



送られてきたhtmlを実行

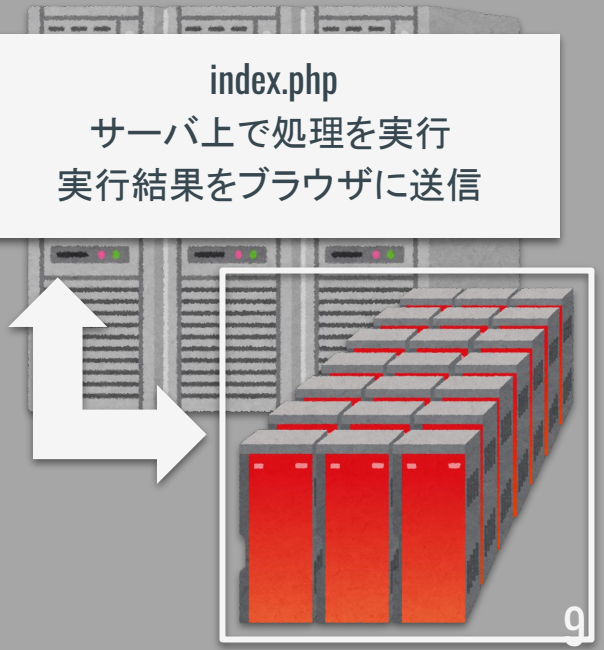
- こういう情報がほしい
- こういう処理をしたい
  例：index.phpにアクセス

http通信

http通信

- 処理した結果のデータ
- 構成したhtml

index.php
サーバ上で処理を実行
実行結果をブラウザに送信



RDBの構造

RDB（Relational Database）とは

複雑なデータが必要なとき

- 一つのテーブルから取得できる情報は限られている.
- ECで購入したユーザの情報と商品の情報を両方取得したい場合は．．？？

複数のテーブル同士を連携（relation）させるッ！！

- テーブルの設計段階で連携させたいidのカラムをつくっておく！
- 両方のテーブルを関連付けるテーブルを「SQLで」作成する！
- 大事なのはデータの構造を見極めること！！

データ構造

One to Many
Many to Many

One to Many（カテゴリテーブルとユーザテーブル）

- 一つのカテゴリには複数のユーザが在籍するが，ユーザの所属は一つ．

category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン

id	name
1	ディオ
2	ツェペリ
3	リサリサ
4	カーズ
5	花京院
6	DIO
7	吉良吉影
8	早人
9	フーゴ
10	ディアボロ
11	エルメス
12	プッチ
13	ジャイロ
14	ヴァレンタイン

One to Many（カテゴリテーブルとユーザテーブル）

- 「多」の側に対応するidを入れる構造にすればOK

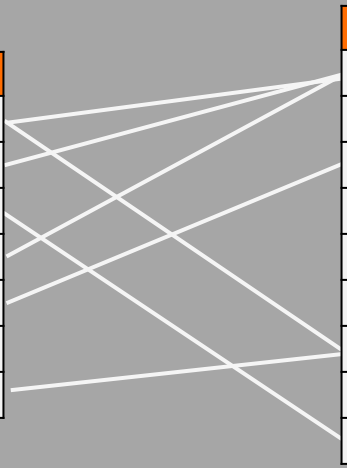
category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン

user_id	name	category_id
1	ディオ	1
2	ツェペリ	1
3	リサリサ	2
4	カーズ	2
5	花京院	3
6	DIO	3
7	吉良吉影	4
8	早人	4
9	フーゴ	5
10	ディアボロ	5
11	エルメス	6
12	プッチ	6
13	ジャイロ	7
14	ヴァレンタイン	7

Many to Many（カテゴリテーブルと擬音語テーブル）

- カテゴリは複数の擬音語が登場し，擬音語も複数のカテゴリで使われる.

category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン



sound_id	name
1	ドドドドド
2	ゴゴゴゴゴ
3	ドォーン
4	バァーン
5	ドグシャア
6	ズキュウゥゥン
7	メメタァ
8	ガオン
9	ズアッ

Many to Many（カテゴリテーブルと擬音語テーブル）

category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン

sound_id	name
1	ドドドドド
2	ゴゴゴゴゴ
3	ドォーン
4	バァーン
5	ドグシャア
6	ズキュウゥゥン
7	メメタァ
8	ガオン
9	ズアッ

何部に

どの音

category_id	sound_id
1	1
1	2
1	7
7	7
3	9
1	6
7	5

このテーブルを「中間テーブル」と呼ぶ

今回やること

前回までに作成したtodoリストにいいね機能（Many to Manyの関係）を追加！

各ユーザは複数のtodoにいいねし，各todoは複数のユーザからいいねされる．

- ユーザテーブルといいねテーブルを作成
- いいねボタン追加
- いいねボタンクリック時の処理
- いいね数を表示

テーブルの追加

【課題2】ユーザ管理機能の作成

ユーザ管理テーブル（<- 必ず作成，DBはこれまでのものを使用）

- テーブル名： users_table
- カラム名など

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
☐	1	id 🔑	int(12)			いいえ	なし		AUTO_INCREMENT	変更 削除 その他
☐	2	username	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			変更 削除 その他
☐	3	password	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			変更 削除 その他
☐	4	is_admin	int(1)			いいえ	なし			変更 削除 その他
☐	5	is_deleted	int(1)			いいえ	なし			変更 削除 その他
☐	6	created_at	datetime			いいえ	なし			変更 削除 その他
☐	7	updated_at	datetime			いいえ	なし			変更 削除 その他

新しくテーブルを作成

いいねテーブル

- テーブル名： like_table
- カラム名など

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
☐	1	id 🔑	int(12)			いいえ	なし		AUTO_INCREMENT	変更 削除 その他
☐	2	user_id	int(12)			いいえ	なし			変更 削除 その他
☐	3	todo_id	int(12)			いいえ	なし			変更 削除 その他
☐	4	created_at	datetime			いいえ	なし			変更 削除 その他

いいねボタンの設置

やりたいこと

いいねボタンをクリックしたら．．．

- like_tableに「誰が」「何に」いいねをしたのかを追加

実装の方針

- todo_read.phpにいいねボタンを設置
- GETでtodoのidとユーザのidを送信する
- 受け取り側のファイル（like_create.php）で受け取ったデータをdbに登録

likeのリンクをつくる（todo_read.php）

```
// ユーザid取得
$user_id = $_SESSION['id'];

// タグ生成部分

$output .= "<td>
<a href='like_create.php?user_id={$user_id}&todo_id={$record["id"]}'>
like</a>
            </td>";

$output .= "<td>      // 編集ボタン
              <a href='todo_edit.php?id={$record["id"]}'>edit</a>
            </td>";
...
// user_idはログイン時にセッション変数に保存している値を使用している.
```

いいねボタンの確認（todo_read.php）

いいねボタンクリックの動作

いいね追加の処理

いいねデータ受取時の処理（like_create.php）

- データの受取
- データベース接続
- テーブルへの登録
- 完了したら一覧画面へ移動

いいねデータ受取時の処理(like_create.php)

```php
// まずはこれ
var_dump($_GET);
exit();

// 関数ファイルの読み込み
include('functions.php');

// GETデータ取得
$user_id = $_GET['user_id'];
$todo_id = $_GET['todo_id'];

// DB接続
$pdo = connect_to_db();
```

いいねデータ受取時の処理(like_create.php)

```
$sql = 'INSERT INTO like_table(id, user_id, todo_id, created_at)
          VALUES(NULL, :user_id, :todo_id, sysdate())'; // SQL作成

$stmt = $pdo->prepare($sql);
$stmt->bindValue(':user_id', $user_id, PDO::PARAM_INT);
$stmt->bindValue(':todo_id', $todo_id, PDO::PARAM_INT);
$status = $stmt->execute();           // SQL実行

if ($status == false) {
  // エラー処理
} else {
  header('Location:todo_read.php');
}
```

いいね追加の動作確認

いいねボタンをクリックして，データが入っていればOK！

←T→	id	user_id	todo_id	created_at
編集 コピー 削除	1	1	10	2020-06-10 11:20:59

ボタン作成 -> 動作確認まで練習！

いいね済かどうかで条件分岐

すでにいいねしていればいいねを取り消す

このままだと. . .

- 連打すれば無限にいいねできてしまう. . .
- いいねしている状況であれば, いいねを取り消す処理にしたい！

実装の方針

- いいねボタンをクリックしたtodoとuserでテーブルを検索
- データが1件以上存在すれば削除のSQLを作成して実行
- データが存在しなければ登録のSQL（前項で作成したもの）を実行

いいねしているかどうかのチェック（like_create.php）

```
// いいね状態のチェック（COUNTで件数を取得できる！）
$sql = 'SELECT COUNT(*) FROM like_table
          WHERE user_id=:user_id AND todo_id=:todo_id';
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':user_id', $user_id, PDO::PARAM_INT);
$stmt->bindValue(':todo_id', $todo_id, PDO::PARAM_INT);
$status = $stmt->execute();
if ($status == false) {
  // エラー処理
} else {
  $like_count = $stmt->fetch();
  var_dump($like_count[0]);       // データの件数を確認しよう！
  exit();
}
```

いいね有無で条件分岐（like_create.php）

```
// いいねしていれば削除，していなければ追加のSQLを作成
if ($like_count[0] != 0) {
  $sql = 'DELETE FROM like_table
            WHERE user_id=:user_id AND todo_id=:todo_id';
} else {
  $sql = 'INSERT INTO like_table(id, user_id, todo_id, created_at)
            VALUES(NULL, :user_id, :todo_id, sysdate())';
}

// INSERTのSQLは前項で使用したものと同じ！
// 以降（SQL実行部分と一覧画面への移動）は変更なし！
// SQL文は1行にまとめる
```

いいね削除の動作確認（like_create.php）

←T→	id	user_id	todo_id	created_at
編集 コピー 削除	1	1	10	2020-06-10 11:34:45
編集 コピー 削除	2	1	16	2020-06-10 11:34:54

いいねしたあとでもう一度クリックし，
データが削除されていればOK！

←T→	id	user_id	todo_id	created_at
編集 コピー 削除	2	1	16	2020-06-10 11:34:54

動作確認までやってみよう！

いいね数の表示

一覧画面のいいねボタン横にいいね数を表示

一覧画面で何件いいねされているかがわからない．．．

- いいねしたらいいねの件数が表示されるように！

実装の方針

- todoごとのいいね数を集計
- 取得したいいね数をtodoのテーブルに結合させる
- いいね数をいいねボタンに表示する

各todoのいいね数を集計

【解説】GROUP BY

GROUP BYを使うと集計ができる！

- 例（phpmyadminで実行しよう！）
- SELECT todo_id, COUNT(id) AS cnt FROM like_table GROUP BY todo_id;
- 「like_table」の「todo_id」ごとに「idの数」を「cnt」というカラム名で表示

注意点

- COUNT()で件数を取得しているが，集計関数以外だとエラーになる
- その他の集計関数
    - MIN(), MAX(), SUM(), ...など

いいね数の集計をしてみる

まずは記述したSQLをphpmyadminで実行しよう！

todo_id	cnt
10	1
15	1
16	1

「どのタスクに」「何件」
いいねがついているかが確認できる！

【Point】このSQL文は集計結果のテーブルを返す！！

テーブルの結合

テーブルを結合して情報をまとめる

集計はできたので，いいねボタンに件数を表示したい！

- 画面の表示はtodo_tableからデータなので，集計結果を組み込めない．．．
- テーブルを結合させることで，集計結果もまとめて表示できる！

todo_read.phpのSQL部分を変更する

- SQLの方針
  - 「todo_table」と「集計結果のテーブル」をつなげる
  - 「todo_tableのid」と「集計結果のテーブルのtodo_id」を対応させる
  - 「集計結果のテーブル」は前項で取得したアレ

結合前

todo_tableとGROUP BYしたテーブル

	id	todo	deadline	created_at	updated_at
編集 コピー 削除	4	良い食材を買う	2020-06-19	2020-06-04 11:47:54	2020-06-05 12:01:56
編集 コピー 削除	7	PHP選手権準備	2020-06-07	2020-06-04 11:48:37	2020-06-04 11:48:37
編集 コピー 削除	10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20
編集 コピー 削除	15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39
編集 コピー 削除	16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12

todo_id	cnt
10	1
15	1
16	1

【解説】JOIN

JOINを使うとテーブルの結合ができる！

- 例
- SELECT * FROM todo_table LEFT OUTER JOIN result_table ON todo_table.id = result_table.todo_id
- 「todo_table」と「result_table」を結合する
- 「ON」の後でどのカラムを対応させるのかを決定する
- 今回は「todo_tableのid」と「result_tableのtodo_id」が対応

【解説】OUTER JOINとINNER JOIN

OUTER JOINは対応するものがない場合NULLで補完される

- 例（上で実行したものと同じ）
- SELECT * FROM todo_table LEFT OUTER JOIN result_table ON todo_table.id = result_table.todo_id

id	todo	deadline	created_at	updated_at	todo_id	cnt
4	良い食材を買う	2020-06-19	2020-06-04 11:47:54	2020-06-05 12:01:56	*NULL*	*NULL*
7	PHP選手権準備	2020-06-07	2020-06-04 11:48:37	2020-06-04 11:48:37	*NULL*	*NULL*
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1

【解説】OUTER JOINとINNER JOIN

INNER JOINは対応するものがない場合は削除される

- 例
- SELECT * FROM todo_table INNER JOIN result_table ON todo_table.id = result_table.todo_id

id	todo	deadline	created_at	updated_at	todo_id	cnt
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1

テーブルを結合して一覧表示（todo_read.php）

```
//データ表示SQL作成
// $sql = 'SELECT * FROM todo_table';     // <- select文を変更
$sql = 'SELECT * FROM todo_table
          LEFT OUTER JOIN (SELECT todo_id, COUNT(id) AS cnt
          FROM like_table GROUP BY todo_id) AS likes
          ON todo_table.id = likes.todo_id';
$stmt = $pdo->prepare($sql);                    // 変更なし
$status = $stmt->execute();                     // 変更なし
```

JOINを使って結合！

まずはSQL文をphpmyadminで実行！

id	todo	deadline	created_at	updated_at	todo_id	cnt
4	良い食材を買う	2020-06-19	2020-06-04 11:47:54	2020-06-05 12:01:56	*NULL*	*NULL*
7	PHP選手権準備	2020-06-07	2020-06-04 11:48:37	2020-06-04 11:48:37	*NULL*	*NULL*
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1

「todo_table」と「集計結果」が結合される！！

集計結果を表示（todo_read.php）

```
// いいねボタン

<a href='like_create.php?user_id={$user_id}&todo_id={$record["id"]}'>
    like{$record["cnt"]}    // cntカラムの数値（いいね数）を追加
</a>
```

テーブルを結合したので,
カラム名「cnt」を指定していいね件数を取れる！！

いいねボタンの表示確認

クリックすると1増える or 1減る

課題

DB連携アプリ(作成 / 参照 / 更新 / 削除 / 集計 / 結合)

DBを使用したアプリを実装しよう！

- DB名 :　　授業で作成したものでOK
- テーブル名 :　　自由に！

卒制のプロトタイプ的な作品とか, SNS的なものとか！

「画面に必要な情報」を決めてそれを元にテーブルを設計するのじゃ！

締切は次回授業前木曜「23:59:59」

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！！