DB update & delete

G's ACADEMY FUKUOKA

アジェンダ

- webの仕組み（復習）
- DB接続の関数化
- PHPからDB操作
  - 更新
  - 削除
- 課題発表→チュータリング(演習)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

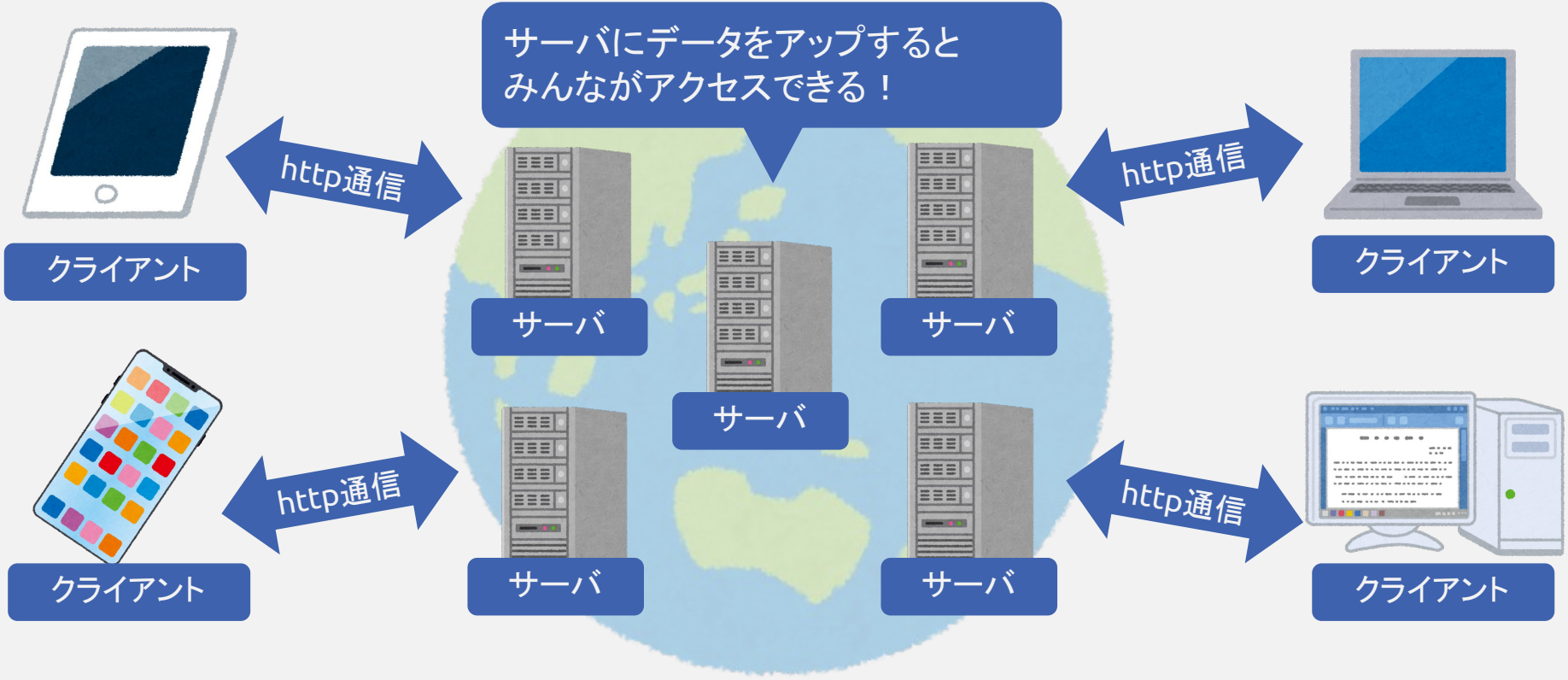
- XAMPPの起動確認
- http://localhost/のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

今日のゴール

- CRUD処理の完成！
- 効率良いコードにする！
- （コード管理を知る）

webの仕組み（復習）

雑なwebの仕組み



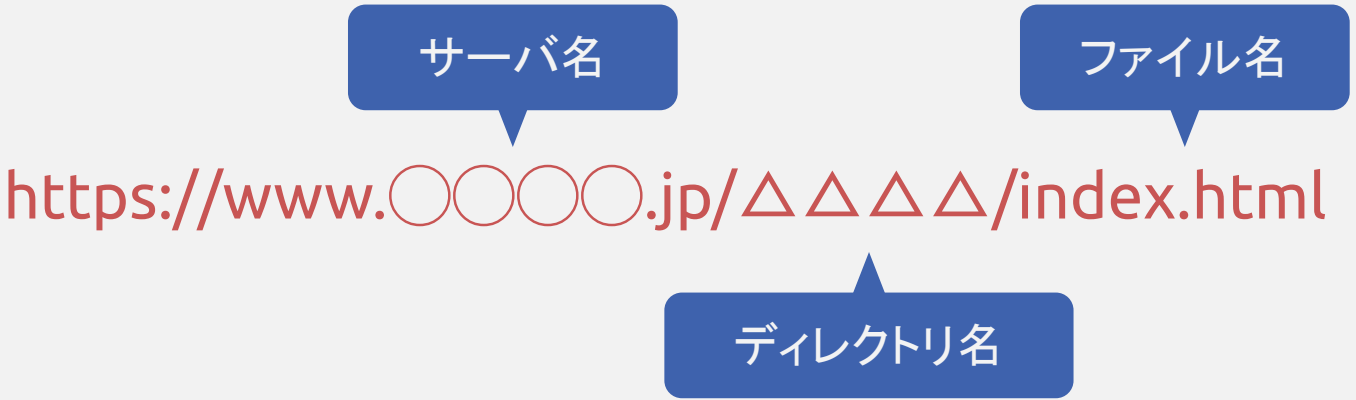
G's ACADEMY FUKUOKA

URL

■URLとは

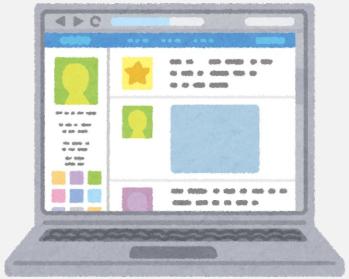
- web上にある情報（ファイル）の場所を指し示す住所.
- Uniform Resource Locatorの略（覚えなくてOK）.

■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

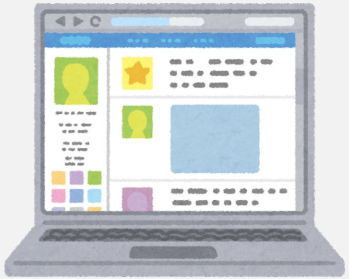
http通信

http通信

- 処理した結果のデータ
- 構成したhtml

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！



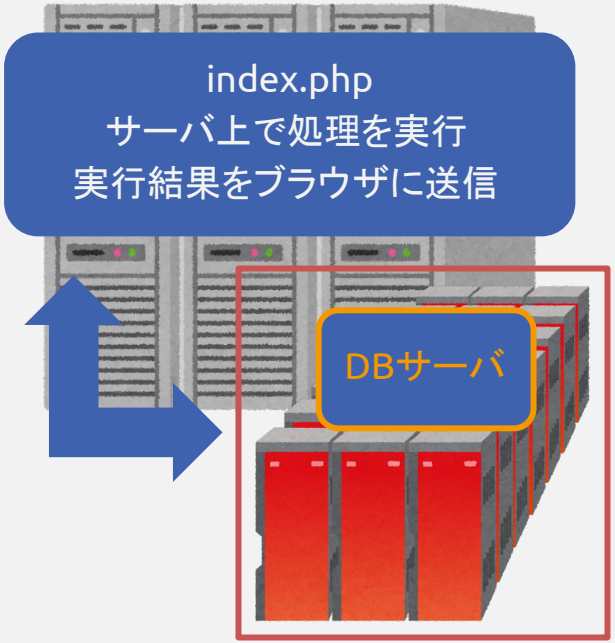
送られてきたhtmlを実行

- 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

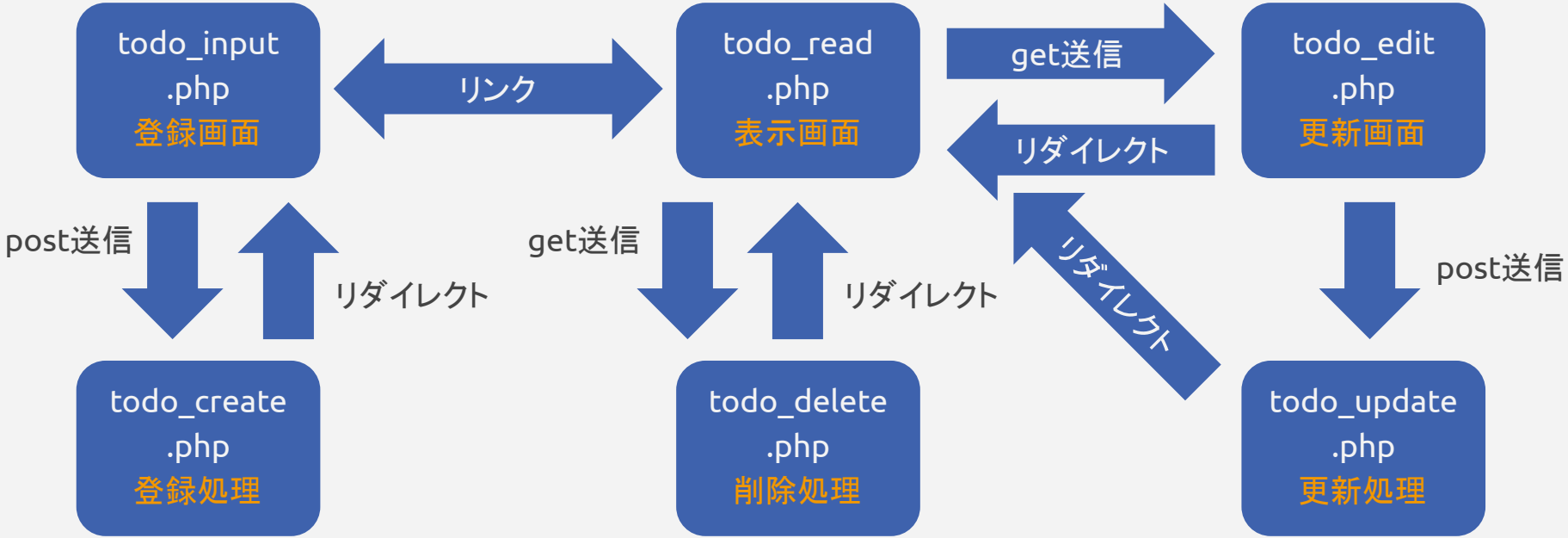
http通信

http通信

- 処理した結果のデータ
- 構成したhtml



todoアプリの全体像



データ更新処理の前に．．．

DB接続は常に同じコード．．！

- 実は．．．
    - todo_input.phpとtodo_read.phpで記述したDB接続のコードは全く同じ！
    - 今回作成するtodo_edit.php, todo_update.php, todo_delete.phpも同じ記述！
- であれば．．！
    - 一つの関数にまとめられる！
    - 関数用のファイルを作成しよう！（functions.php）

DB接続関数を定義！

functions.php

```
// 関数の定義
function connect_to_db(){
  $dbn='mysql:dbname=YOUR_DB_NAME;charset=utf8;port=3306;host=localhost';
  $user = 'root';
  $pwd = '';
  try {
      return new PDO($dbn, $user, $pwd);
  } catch (PDOException $e) {
      exit('dbError:'.$e->getMessage());
  }
}
```

DB接続が必要なファイルで呼び出し&実行！

todo_create.php

```
// 呼び出し（todo_create.php, toodo_read.php, など）

include('functions.php');          // 関数を記述したファイルの読み込み
$pdo = connect_to_db();            // 関数実行
```

データ更新処理

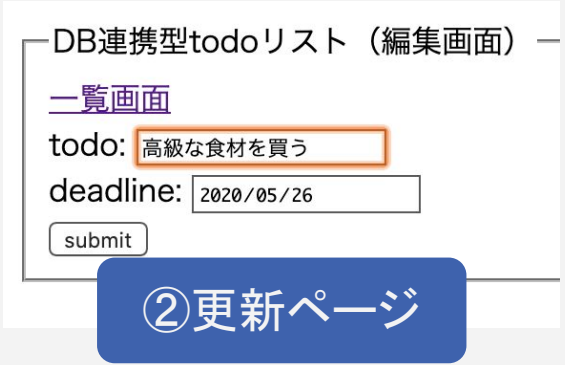
SQL構文：UPDATE（データ更新）

```
// 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;
// 必ずWHEREを使用！！→忘れると全てのデータが更新されます．．！
```

データ更新処理

- 更新処理の流れ（登録処理と似ています！）
  - ① 一覧画面に更新ページへのリンクを作成
    - urlにidを追加：todo_edit.php?id=**
  - ② 更新ページの作成（todo_edit.php）
  - ③ 更新処理の作成（todo_update.php）
  - ④ 一覧画面に戻る



①一覧画面に更新ページへのリンクを追加

todo_read.php

```
// 出力部分を下記に変更
foreach ($result as $record) {
  $output .= "<tr>";
  $output .= "<td>{$record["deadline"]}</td>";
  $output .= "<td>{$record["todo"]}</td>";
  // edit deleteリンクを追加
  $output .= "<td><a href='todo_edit.php?id={$record["id"]}'>edit</a></td>";
  $output .= "<td>
              <a href='todo_delete.php?id={$record["id"]}'>delete</a></td>";
  $output .= "</tr>";
}
```

①動作確認（一覧画面）

todo_read.php

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	edit	delete
2020-05-27	お酒を買う	edit	delete

editボタンにカーソルを載せて
「?id=**」になればOK！

localhost/php_tutorial/php03/php03_comp/todo_edit.php?id=1

②更新ページを作成

todo_edit.php

```php
// 関数ファイル読み込み
include("functions.php");

// 送信されたidをgetで受け取る
$id = $_GET['id'];

// DB接続&id名でテーブルから検索
$pdo = connect_to_db();
$sql = 'SELECT * FROM todo_table WHERE id=:id';
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':id', $id, PDO::PARAM_INT);
$status = $stmt->execute();
```

1つのデータを取得するためにidでDBを検索する！

②更新ページを作成

todo_edit.php

```
// fetch()で1レコード取得できる.
if ($status == false) {
  $error = $stmt->errorInfo();
  echo json_encode(["error_msg" => "{$error[2]}"]);
  exit();
} else {
  $record = $stmt->fetch(PDO::FETCH_ASSOC);
}
...
// htmlのタグに初期値として設定
<input type="text" name="todo" value="<?= $record["todo"] ?>">
<input type="date" name="deadline" value="<?= $record["deadline"] ?>">
```

②更新ページを作成

todo_edit.php

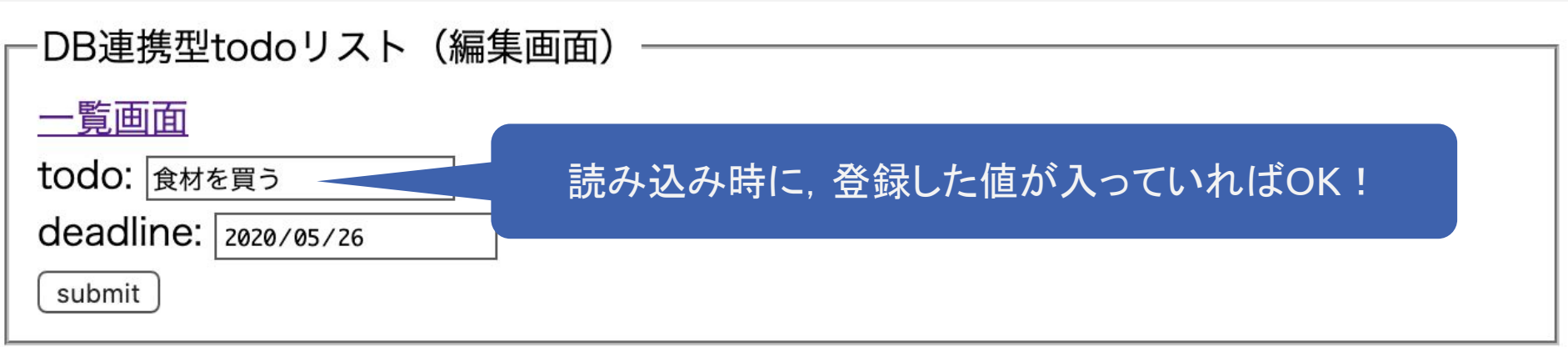
```
// idを見えないように送る
//  - input type="hidden"を使用する！

// form内に以下を追加
<input type="hidden" name="id" value="<?=$record['id']?>">

// 更新のformは，登録と同じくpostで各値を送信しています！
```

②動作確認（更新ページ）

todo_edit.php



G's ACADEMY FUKUOKA

③更新処理を作成 ④リダイレクト

todo_update.php

```
// 各値をpostで受け取る
$id = $_POST['id'];
...

// idを指定して更新するSQLを作成
$sql = "UPDATE todo_table SET todo=:todo, deadline=:deadline,
            updated_at=sysdate() WHERE id=:id";
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':todo', $todo, PDO::PARAM_STR);
$stmt->bindValue(':deadline', $deadline, PDO::PARAM_STR);
$stmt->bindValue(':id', $id, PDO::PARAM_INT);
$status = $stmt->execute();
```

③更新処理を作成 ④リダイレクト

todo_update.php

```
// 各値をpostで受け取る
if ($status == false) {
  // SQL実行に失敗した場合はここでエラーを出力し，以降の処理を中止する
  $error = $stmt->errorInfo();
  echo json_encode(["error_msg" => "{$error[2]}"]);
  exit();
} else {
  // 正常に実行された場合は一覧ページファイルに移動し，処理を実行する
  header("Location:todo_read.php");
  exit();
}
```

データ削除処理

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！！
```

削除処理

- 削除処理の流れ

済 ① 一覧画面に削除ページへのリンクを作成

urlにidを追加todo_delete.php?id=**

② 削除処理の作成（todo_delete.php）

③ 一覧画面に戻る

②削除処理を作成 ③リダイレクト

todo_delete.php

```php
// idをgetで受け取る
$id = $_GET['id'];

// idを指定して更新するSQLを作成
'DELETE FROM todo_table WHERE id=:id'

// 一覧画面にリダイレクト
header('Location:todo_read.php');
```

動作確認（削除処理）

todo_read.php

G's ACADEMY FUKUOKA

課題

【課題1】DB連携アプリ（更新処理 / 削除処理）

- 前回の課題に機能追加！

- 下記ファイル（処理）を追加しよう！
  - 表示処理　←更新画面へのリンクなど追加
  - 更新画面
  - 更新処理
  - 削除処理

- もちろん新しいアイデアで作り直してもOK！
- そろそろwebアプリケーションの全体（卒制含め）が実装できるぞッ！

【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	user_id	varchar(128)	utf8_unicode_ci		いいえ	なし			変更 削除 その他
☐	3	password	varchar(128)	utf8_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

【課題2】ユーザ管理機能の作成

- 前スライドでつくったユーザのデータを管理する処理を実装！
  - ユーザ追加処理
  - ユーザー覧表示処理
  - ユーザデータ更新処理
  - ユーザデータ削除処理
  - （サービス管理者がユーザのデータを操作するイメージ）
- 課題1と課題2はそれぞれ独立でOK！

提出は次回授業前まで！！

Githubタイム！

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！！