session

G's ACADEMY
FUKUOKA

アジェンダ

- webの仕組み（復習）
- セッション機能
- 認証処理の実装
  - ログイン処理
  - ログアウト処理
- 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

- ページ間でデータ共有する方法を知る！
- データの管理方法を学ぶ！
- ログイン&ログアウト処理を実装する！

前回の課題

【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

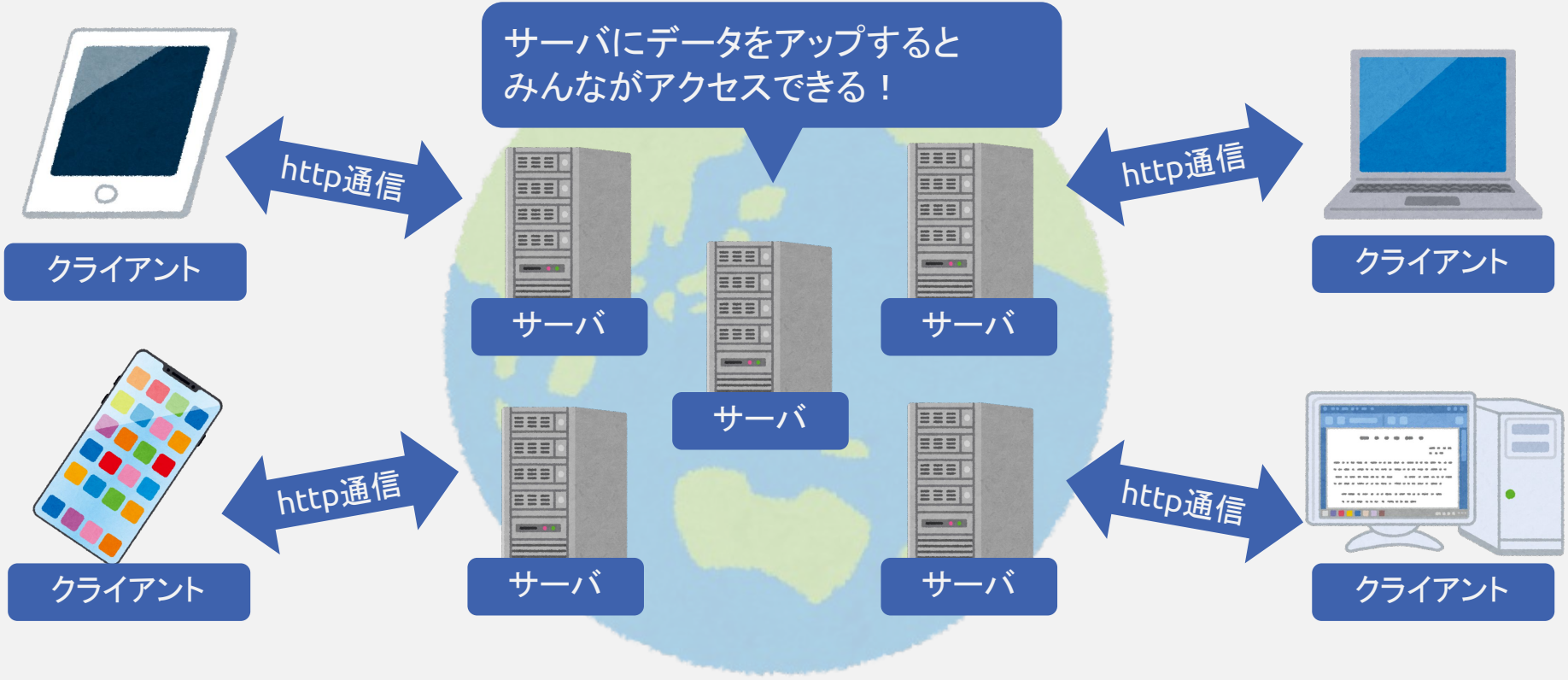
	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
☐	1	id	int(12)			いいえ	なし		AUTO_INCREMENT	変更 削除 その他
☐	2	username	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			変更 削除 その他
☐	3	password	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			変更 削除 その他
☐	4	is_admin	int(1)			いいえ	なし			変更 削除 その他
☐	5	is_deleted	int(1)			いいえ	なし			変更 削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

【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！

webの仕組み（復習）

雑なwebの仕組み

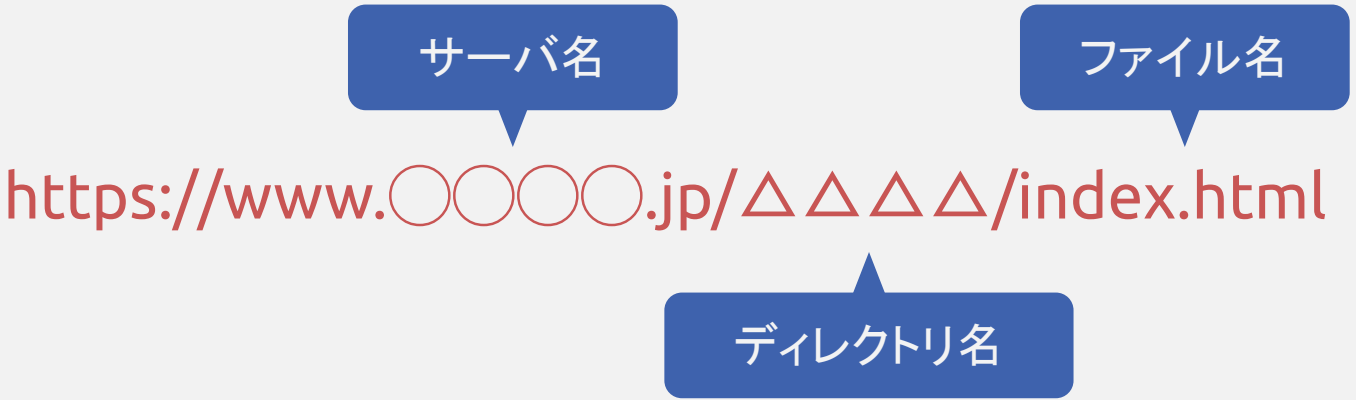

サーバにデータをアップすると
みんながアクセスできる！
http通信
http通信
http通信
http通信
クライアント
クライアント
クライアント
クライアント
サーバ
サーバ
サーバ
サーバ
サーバ

URL

■URLとは

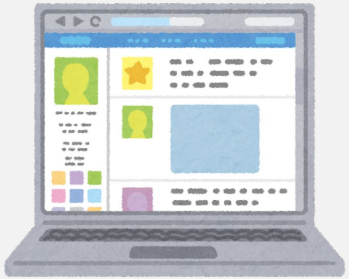
- web上にある情報（ファイル）の場所を指し示す住所.
- Uniform Resource Locatorの略（覚えなくてOK）.

■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

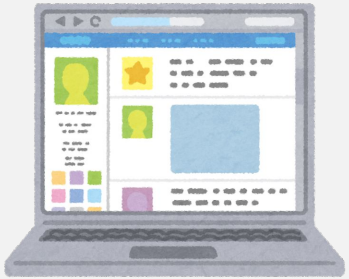
http通信

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！



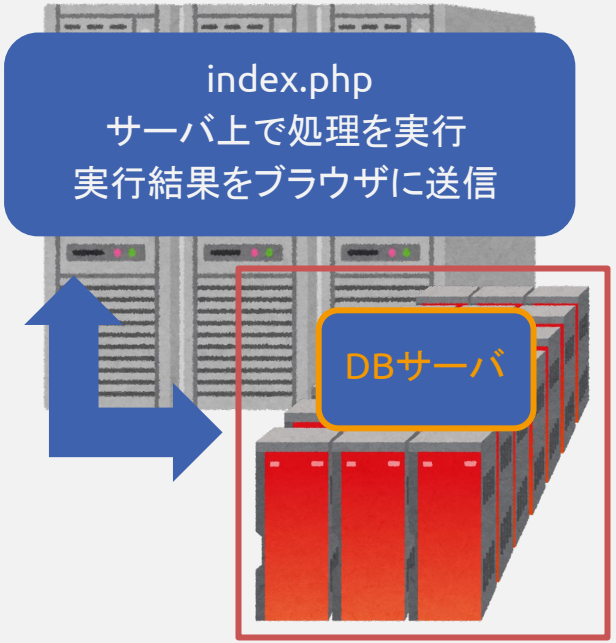
送られてきたhtmlを実行

- 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

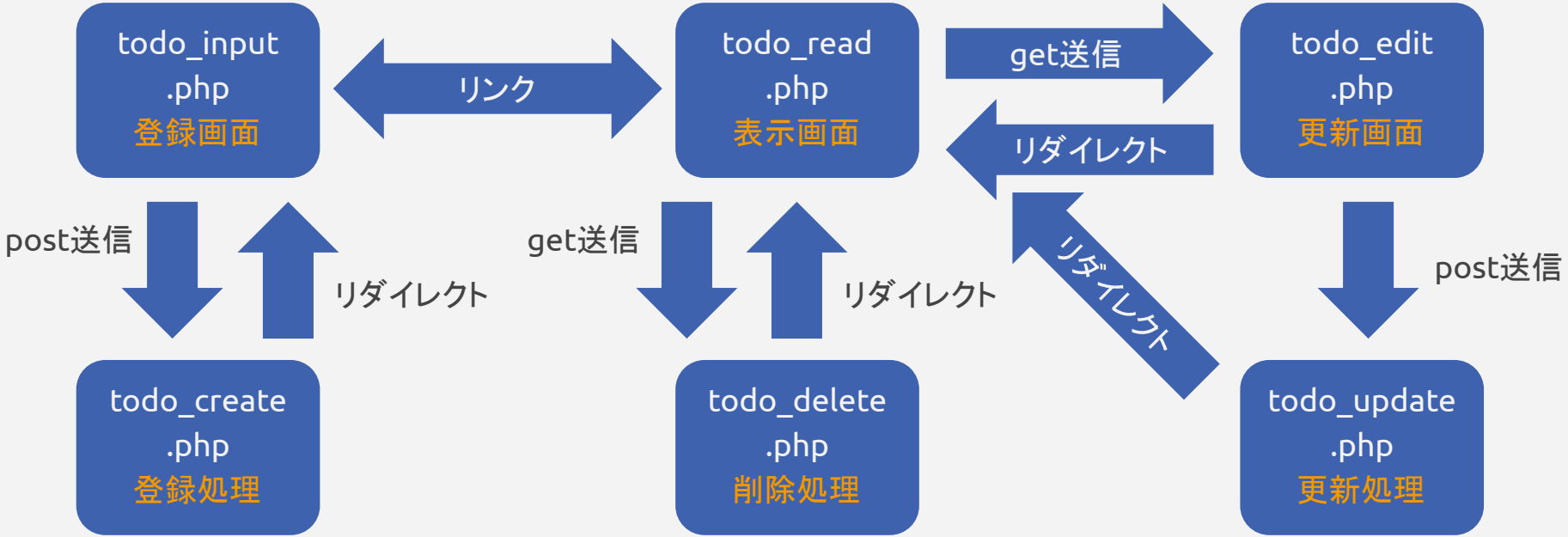
http通信

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

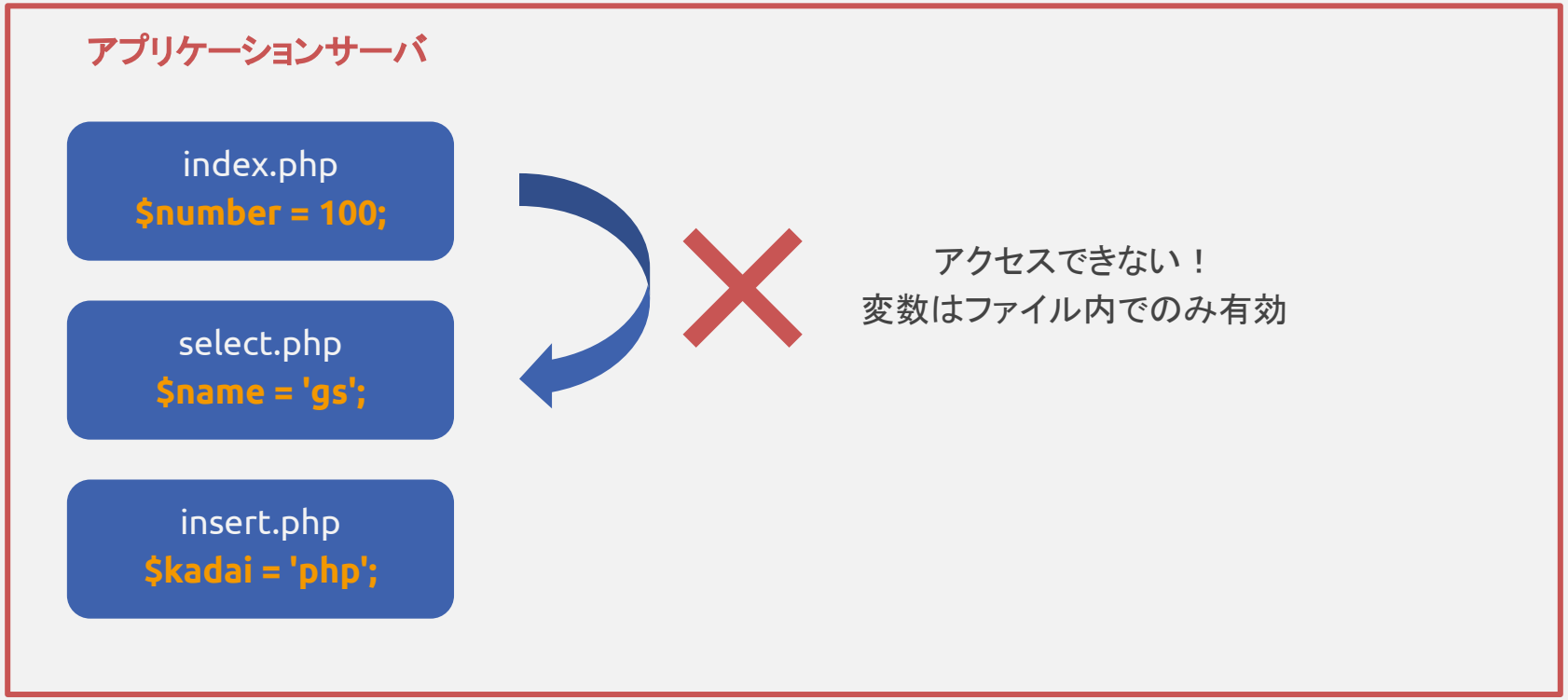


セッション機能

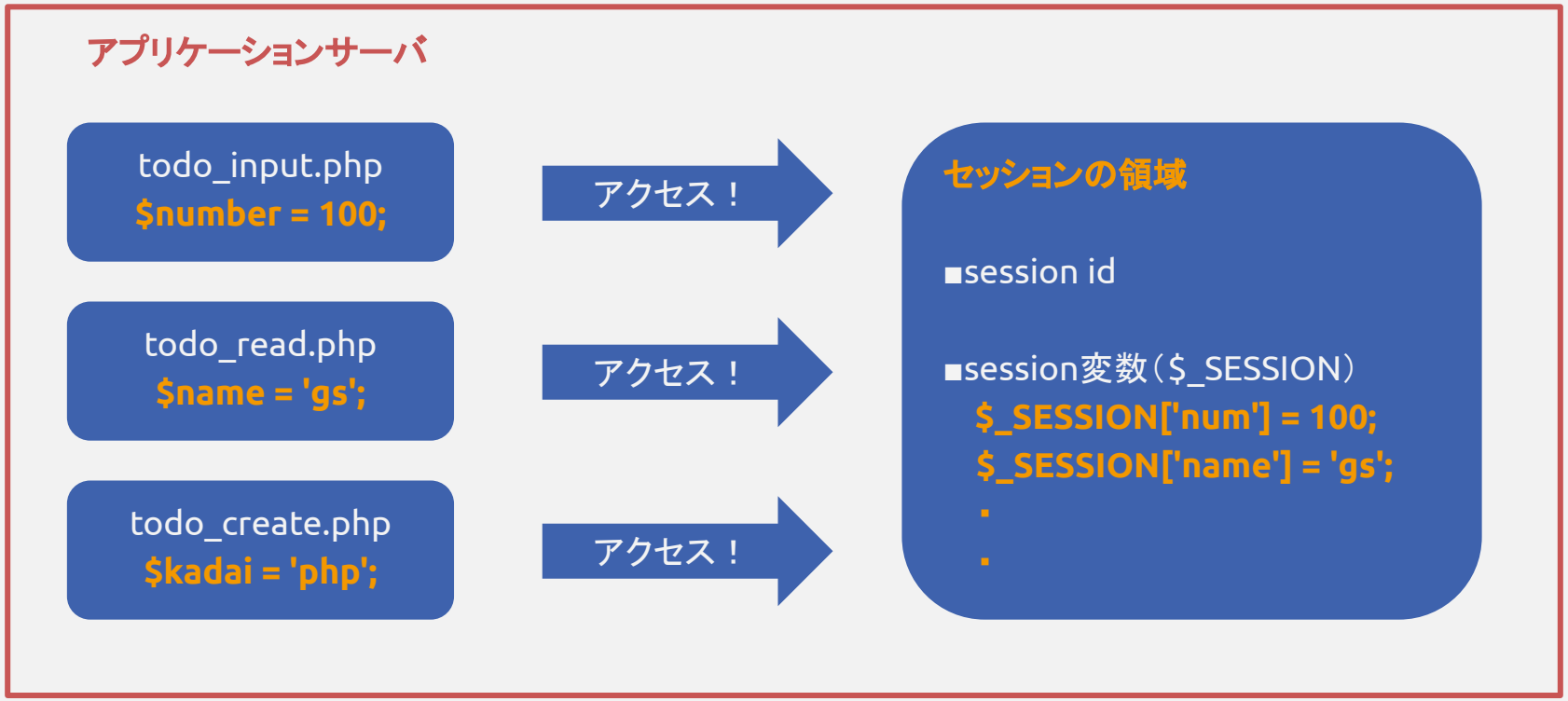
sessionとは．．？

- 何か？？
  - サーバに変数などを保存できる仕組み.
    ……以上である！
- 補足
  - サーバ自体に変数を定義する.
  - サーバ上にあるどのファイルからでも値を取り出せる！

sessionとは．．？


アプリケーションサーバ
index.php
$number = 100;
select.php
$name = 'gs';
insert.php
$kadai = 'php';
アクセスできない！
変数はファイル内でのみ有効

sessionとは．．？


アプリケーションサーバ
todo_input.php
$number = 100;
todo_read.php
$name = 'gs';
todo_create.php
$kadai = 'php';
アクセス！
アクセス！
アクセス！
セッションの領域
■session id
■session変数（$_SESSION）
$_SESSION['num'] = 100;
$_SESSION['name'] = 'gs';

session_id

sessionとは．．？

- sessionを始める
  - sessionがスタートすると「session領域」が作られる.
  - session領域識別用のid（session_id）が発行される.
  - sessionの機能が使えるようになり，情報を保存できる.
- session_idの再生成ができる
  - 悪意あるサイトにsession_idを読まれてしまうとハッキングのリスク.
- sessionを終了する
  - 保存されている情報などを破棄する.

sessionでやることの流れ

- sessionのスタート
    - session_idの管理，必要に応じて$_SESSION（session変数）にデータを保存.
- session_idの再生成
    - ページ移動などのタイミングでidを更新する.
- sessionの終了

sessionでやることの流れ

session_id.php

-まずはsessionを宣言

- session_start();　←使用するファイルでは必ず最初に記述する！
- idが発行されてブラウザにidが保存される.

- idの確認方法
- ①PHPファイル上でsession_id();で取得可能.
- ②ブラウザで「検証→Application→Cookies→localhost」

session_idの再生成（セキュリティ）

session_regenerate_id.php

- session_regenerate_id();
  - sessionのidがバレると他の人にsessionの中身をいじられてしまう可能性．．！
  - session_regenerate_id();を使用するとidを再生成して更新できる.
  - （保存されているデータ自体は変更なし）
- 使い所
  - ログインしたらid発行してログイン情報を管理.
  - ページ移動したタイミングで再生成&更新

idの管理と再生成

session_regenerate_id.php

```php
// session_regenerate_idの例
<?php
session_start();                                    // セッション開始
$old_session_id = session_id();                     // idの取得
session_regenerate_id(true);                        // id再生成&旧idを破棄
$new_session_id = session_id();                     // 新idの取得
echo '<p>旧id' . $old_session_id . '</p>';// idの確認
echo '<p>新id' . $new_session_id . '</p>';
?>
```

idの管理と再生成

session_regenerate_id.php

- 練習
  - idを発行して確認しよう！
  - 再生成して旧idと新idを表示しよう！

session変数

session変数を使う！

session01.php

```
// サーバに変数を保存する！
// $_SESSION['変数名']で宣言.

// 例
<?php
session_start();                    // session変数を使用する場合も必須！
$_SESSION['num'] = 100;             // session変数の宣言
echo $_SESSION['num'];
?>
```

session変数を使う！

session02.php

```
// サーバに保存されている変数を取り出す.

// 例
<?php
session_start();                    // 必須！
$_SESSION['num'] += 1;          // session変数を+1する
echo $_SESSION['num'];          // 結果を出力
?>

// session02.phpでは変数を定義していないが，セッションの機能で呼び出せる！
```

session変数を使う！

session02.php

- 練習
  - session01.phpでsession変数を定義しよう！
  - session02.phpで定義した変数を呼び出して出力しよう！

sessionの終了

sessionの終了（情報の破棄）

```
// session変数の削除
unset($_SESSION[key]);    // 該当するsession変数を削除

// session情報の全削除
$_SESSION = array();           // 情報を全て削除
setcookie(session_name(), '', time() - 42000, '/');

// sessionを破壊
session_destroy();

// 参考：https://www.php.net/manual/ja/function.session-destroy.php
```

認証処理

認証（ログイン&ログアウト）の全体像

- 必要なファイル
  - todo_login.php　　ログイン情報（id, pwd）を入力して送信
  - todo_login_act.php　　送信されたデータを受け取り，DB関連の処理を実行
  - todo_logout.php　　セッション，ログイン情報の破棄

ログイン処理

ログイン処理の流れ

- ログイン
  - ①ログインフォーム情報を入力して送信（todo_login.php）
  - ②送信されたデータを受け取る（todo_login_act.php）
  - ③受け取ったデータがDBにあるかどうかチェック（todo_login_act.php）

ログイン処理の流れ

- 成功時（DBにユーザのデータが存在した場合）
    - ①DBにログイン情報があればセッション変数に格納（todo_login_act.php）
    - ②セッション変数にログイン情報を保持してtodo_read.phpに移動
- 失敗時（DBにユーザのデータが存在しなかった場合）
    - ①DBにログイン情報がなければtodo_login.phpに戻る（ログイン失敗）

ログイン処理の流れ

todo_login.php

```
<form action="todo_login_act.php" method="POST">
  ...
    <div>
      username: <input type="text" name="username">
    </div>
    <div>
      password: <input type="text" name="password">
    </div>
    <div>
      <button>Login</button>
    </div>
  ...
</form>
```

ログイン処理の流れ

todo_login_act.php

```
// セッション開始&ログイン情報の受け取り
<?php
session_start();                    // セッションの開始
include('functions.php');           // 関数ファイル読み込み
$pdo = connectToDb();               // DB接続
$username = $_POST[username];       // データ受け取り→変数に入れる
$password = $_POST['password'];
...
```

ログイン処理の流れ

todo_login_act.php

```
// DBにデータがあるかどうか検索
$sql = 'SELECT * FROM users_table
          WHERE username=:username
            AND password=:password
            AND is_deleted=0';

$stmt = $pdo->prepare($sql);
$stmt->bindValue(':username', $username, PDO::PARAM_STR);
$stmt->bindValue(':password', $password, PDO::PARAM_STR);
$status = $stmt->execute();
```

WHEREで条件を指定！

ログイン処理の流れ

todo_login_act.php

```
// DBのデータ有無で条件分岐
$val = $stmt->fetch(PDO::FETCH_ASSOC);    // 該当レコードだけ取得
if (!$val) {
  echo "<p>ログイン情報に誤りがあります．</p>";
  echo '<a href="todo_login.php">login</a>';
  exit();
}
...
```

ログイン処理の流れ

todo_login_act.php

```
// DBにデータがあればセッション変数に格納
} else {
  $_SESSION = array();                    // セッション変数を空にする
  $_SESSION["session_id"] = session_id();
  $_SESSION["is_admin"] = $val["is_admin"];
  $_SESSION["username"] = $val["username"];
  header("Location:todo_read.php");       // 一覧ページへ移動
  exit();
}
```

ログアウト処理

ログアウト（session終了と同じ）

todo_logout.php

```
// セッションの破棄→ログイン画面へ移動
<?php
session_start();                          // セッションの開始
$_SESSION = array();                      // セッション変数を空の配列で上書き
if (isset($_COOKIE[session_name()])) {
  setcookie(session_name(), '', time()-42000, '/');
}                                         // クッキーの保持期限を過去にする
session_destroy();                      // セッションの破棄
header('Location:todo_login.php');// ログインページへ移動
exit();
```

ログイン状態のチェック

ログイン状態のチェック

- ログインしているときのみアクセスできるように！
  - 登録画面，一覧画面などはログインしているユーザのみ見られるようにしたい
  - ログインをチェックし，ログインしていない状態ならログイン画面に移動
  - 「session_idを持っていない」or「idが古い」はログインしていない状態
  - 複数のファイルでチェックを行うため，関数ファイルに記述しよう！

ログイン状態のチェック

functions.php

```
// ログインしているかどうかのチェック→毎回id再生成
function check_session_id () {
  // 失敗時はログイン画面に戻る（セッションidがないor一致しない）
  if (!isset($_SESSION['session_id']) ||
      $_SESSION['session_id']!=session_id()) {
    header('Location: todo_login.php');       // ログイン画面へ移動
  } else {
    session_regenerate_id(true);              // セッションidの再生成
    $_SESSION['session_id'] = session_id();   // セッション変数に格納
  }
}
```

ログイン状態のチェック

```
// 各ページ読み込み時にログインチェック
<?php
session_start();                    // セッションの開始
include('functions.php');           // 関数ファイル読み込み
check_session_id();                 // idチェック関数の実行
...

// 下記ファイルで実施
// todo_input.php, todo_read.php, todo_edit.php, todo_create.php
// todo_create.php, todo_update.php, todo_delete.php
// ログインしていない状態でアクセスしようとするとログインページへ移動
```

課題

これまでのアプリにログイン機能を追加

- 下記処理を追加しよう！
  - ログイン画面
  - ログイン処理
  - ログアウト処理

- ユーザの種類によって権限を分けてみよう
  - ログインしていないユーザ　→情報を見るだけ（登録，更新&削除不可）
  - 一般ユーザ　→情報の登録，表示，更新，削除が可能
  - 管理者ユーザ　→ユーザの登録，表示，更新，削除も可能

例（todoアプリの場合）

- ログインしていないユーザがアクセス可能な画面
  - ログイン画面
  - 編集不可のtodo一覧画面

例（todoアプリの場合）

- 一般ユーザがアクセス可能な画面
  - ログイン画面
  - 編集不可のtodo一覧画面
  - todo一覧画面
  - todo登録画面
  - todo編集画面

例（todoアプリの場合）

- 管理者ユーザがアクセス可能な画面
  - ログイン画面
  - 編集不可のtodo一覧画面
  - todo一覧画面
  - todo登録画面
  - todo編集画面
  - ユーザー覧画面
  - ユーザ編集画面
  - ユーザ編集画面

これは．．．もうwebサービスつくれるぞ！！

提出は次回授業前木曜「23:59:59」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！！