

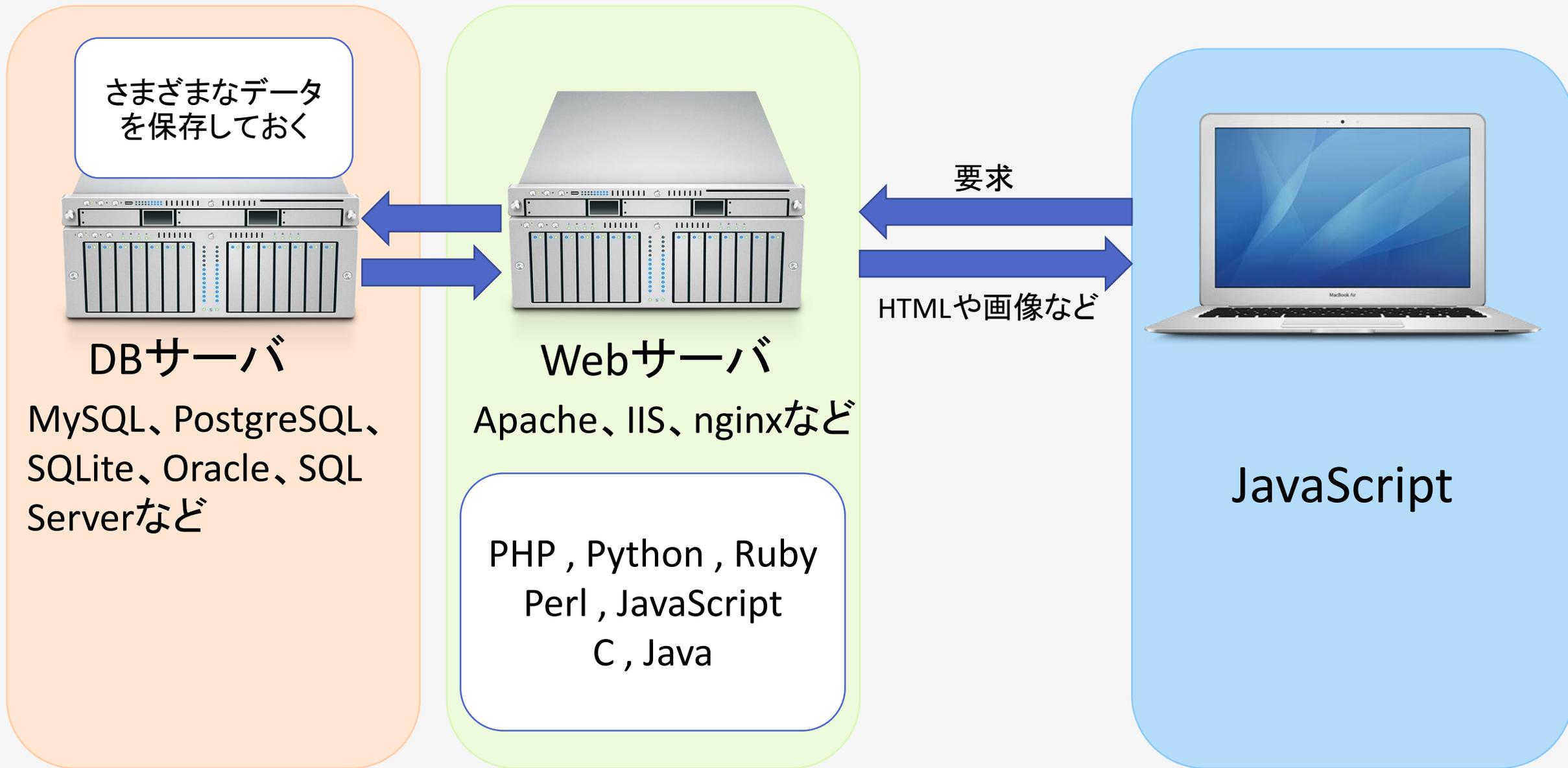
実践課題制作

2019/06/10

Kazuma Sekiguchi

class@cieds.jp

サーバサイドとクライアントサイド



データベース

- 動的に生成するためには、生成するための情報を保存しておく必要がある
 - 一般的には「ファイル」
 - サーバーサイドでもファイルとしてデータを保存しておくことは可能
- ファイルの場合、データが取り出しにくい
 - どこにファイルがあるのか？
 - ファイルの中にどのような形式で記述されているのか？
- データベースという仕組みを利用した方が楽

データベースを利用する理由

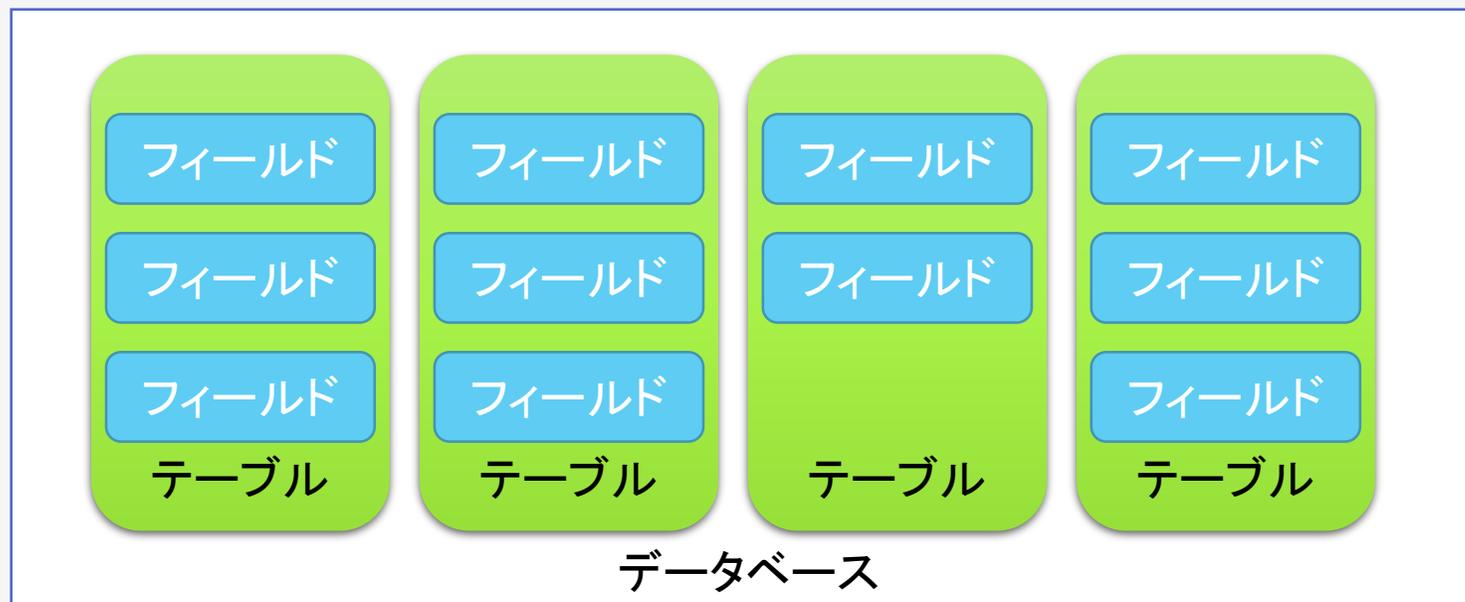
- ファイルに保存することで、データの保存は可能
 - ただし、任意のデータを取り出すには不向き
 - Ex:誕生日が1995年以降の人だけ取り出す
 - すべてのファイルを検索して見つけ出す必要がある→時間が掛かりすぎる
- データベースであれば、内部で特殊形式によりデータを保持
 - 検索時に条件を使用してデータを取り出すことが可能
 - 高速にデータ取り出し、更新が可能
 - 多量のデータも規則性に従って保持可能

データベース（1）

- データをある規則に基づいて保存し、再利用する仕組み
- 大量のデータを保存、変更、削除、取り出すのに優れる
- 基本的に保存できるのは、数値か文字のデータ
 - 画像なども保存可能だが、通常しない

データベース（2）

- データベースは2つの意味がある
 - 1. データベースソフト
 - 2. データベースソフト内で動くテーブルの集合体
- データベースは複数のテーブルから構成される
- テーブルには、いくつかのフィールドが存在する



データベース (3)

- 各テーブルは表形式になっている
- 1つのデータを1行に収める
- 各フィールドに何を収めるかは別に規定をする (数値? 文字? 日付?)

1つのデータ →

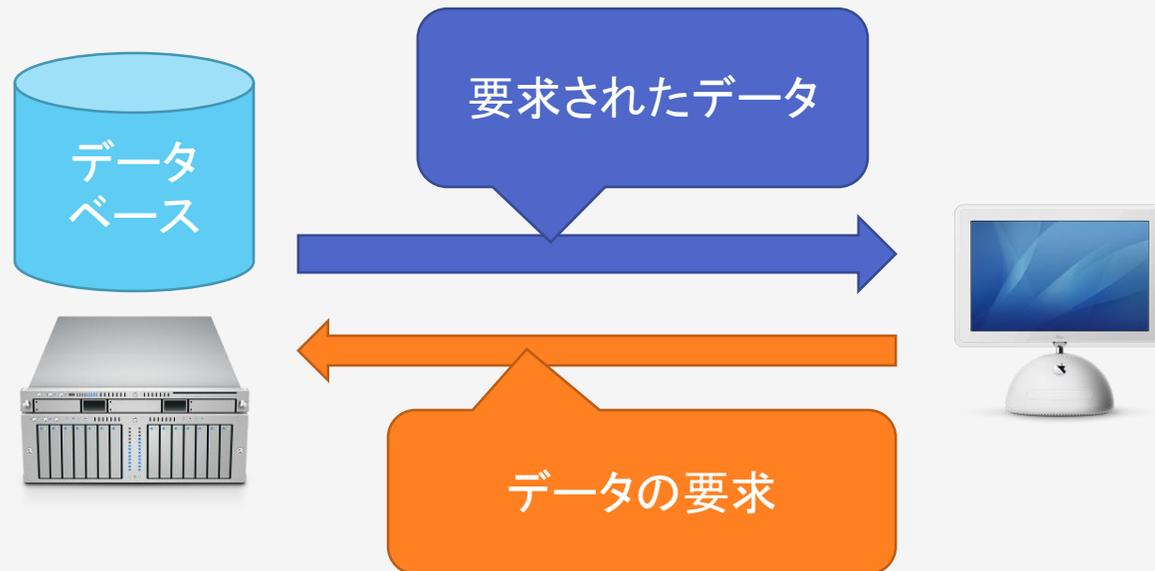
1	みかん	和歌山	100	2008/11/20
2	りんご	青森	150	2008/11/22
3	桃	山梨	320	2008/08/20
4	いちご	三重	80	2008/05/10

データベース（4）

- 各フィールドには型及び長さがある
 - 文字列、数値、日時など
 - 「長さ」はデータの最大の長さを指定する
 - 郵便番号なら「7」など
 - 入れる予定のデータに合わせて形式および長さを選択する必要がある
- 各データを一意に決めるデータが必須
 - 他のデータと必ず違う値を取るフィールドが必要
 - 名前などでも一意になるとは限らない
 - 「ID」というフィールドを作り、順番に数字を付けていくことが多い（データベースに順番に数字を振る機能がある）

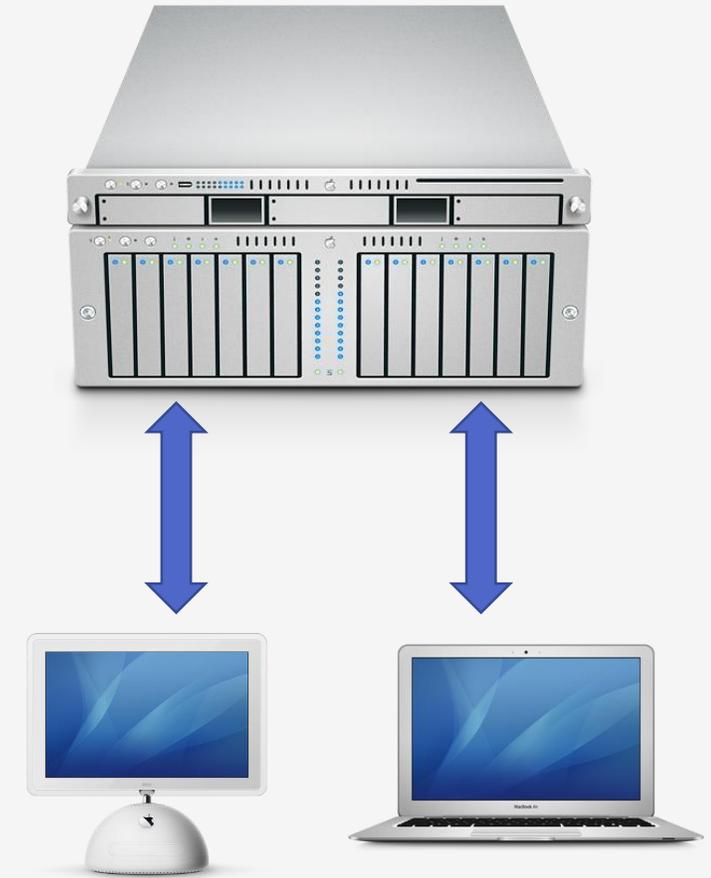
データベース（5）

- ほとんどのデータベースがクライアントサーバ方式をとる
 - データはデータベースサーバにあり、必要とするPCからデータベースサーバに接続して取得する



クライアントサーバ方式

- データベースはサーバソフトとクライアントソフトから構成されるのが一般的
- メリット
 - 複数台から接続し、データを取得してもらえ
 - データベースサーバの数を減らし、データの管理がしやすい
- 中小規模ならば、クライアントとサーバを同じマシンで兼ねることが可能
 - 自分から自分のマシンにあるデータベースサーバに接続する



SQL

Control database



SQL (1)

- データをどのように取得するか？
 - データベースを操作する言語がある
 - SQL (エスキューエル)
 - データベースを操作し、データを挿入、取得、更新、削除を行うための言語
 - 標準化されており、どのデータベースでもほとんど利用可能
(多少の差異はある)
 - SQL自体は小文字でもOKだが、PHPで記述するときは大文字での記述が多い
- SQL文で文字を扱うときは「'」で括る
- 条件式で列名を指定するときも「`」で括る

SQL (2)

- データの取得 (1)

SELECT 「取得するフィールド名」 FROM 「テーブル名」

全部取得

SELECT * FROM 「テーブル名」

条件にあったものだけ取得

SELECT * FROM 「テーブル名」 WHERE `id` = 1

指定した数だけ取得

SELECT * FROM 「テーブル名」 LIMIT 10

SQL (2)

- データの取得 (2)

並び替えて取得

IDが大きいもの順に取得

```
SELECT * FROM 「テーブル名」 ORDER BY id DESC
```

IDが小さいもの順に取得

```
SELECT * FROM 「テーブル名」 ORDER BY id ASC
```

- 複数の条件を満たしたもののだけ取得するには「AND」使用
- 条件のいずれかを満たしたもののだけ取得するには「OR」を使用

SQL (2)

- データの取得 (3)

全部使う

```
SELECT id , name FROM 「テーブル名」 WHERE  
`id` = 1 OR `id` = 10 LIMIT 10 ORDER BY id DESC
```

SQL (3)

- データの更新

UPDATE 「テーブル名」 SET 「フィールド名」 = 「更新する値」

全部更新

UPDATE 「テーブル名」 SET `name` = 'Kazuma'

* テーブル内の全てのnameが変わることに注意！

条件を指定して更新

UPDATE 「テーブル名」 SET `name` = 'Kazuma' WHERE `id` = 1

* ほとんどこの書式を使う

SQL (4)

- データの挿入 (1)

`INSERT INTO 「テーブル名」 VALUES (「挿入する値」)`

- 挿入する値は「,」で区切ることで複数挿入可能
- テーブルで指定されているフィールドの左端の値から指定する
- 値を入力したくないフィールドは「' '」で飛ばす

SQL (4)

- データの挿入 (2)

```
INSERT INTO 「テーブル名」 VALUES  
(1,'kazuma','19800807','kazuma@example.co.jp','')
```

SQL (5)

- データの削除

DELETE FROM 「テーブル名」

* テーブル内のデータが全部消えることに注意！

条件を指定して削除

DELETE FROM 「テーブル名」 WHERE `id` = 1

* ほとんどこの書式を使う

データベースをGUIで作業する

- 通常データベースはコンソールと呼ばれるCUI画面から操作
 - 慣れると便利
 - 慣れるまでは非常に難しい
- GUIで操作するためのツールがある
 - phpmyadmin
 - PHPで書かれたMySQLを操作するためのツール
 - phpPgAdmin
 - PHPで書かれたPostgreSQLを操作するためのツール

主なデータベース（無償）

- MySQL
 - Sun Microsystemsが発売しているデータベース。ライセンスにより、商用版、無償版が存在する。PHPと組み合わせて利用する例が多い。PHP4ではデフォルトのデータベース。
- PostgreSQL
 - 無償で利用可能なデータベース。商用に遜色のない性能と機能を有している。
- SQLite
 - 他とは異なりサーバ形式をとっていないデータベース。PHP5からデフォルトのデータベースに指定された。小規模の環境なら十分対応できる性能を有している。

Create Dynamic Web Page

How about creating dynamic?



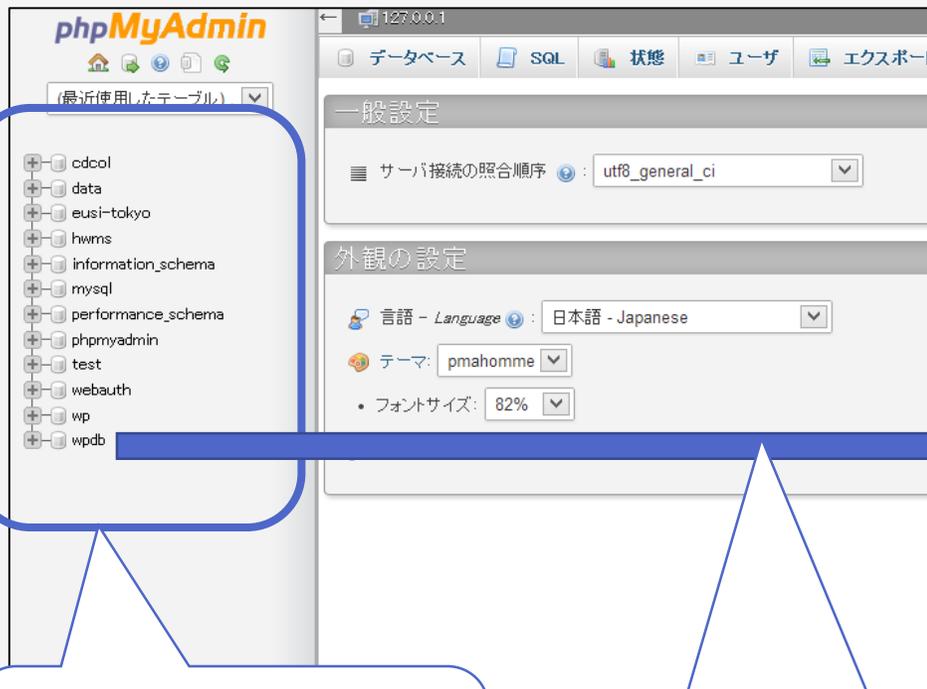
ダイナミックページの作成

- あらかじめDBなどに保存してあるデータを取得してきて表示する
 - 表示する際に条件に応じてデータを取り出してくる
- DBに保存したり更新したりする仕組みが必要
 - 挿入、抽出（取り出し）、更新、削除が必要
 - 実際には削除はしない（削除フラグなどを利用して、表示しないようにするなどしておく）
 - 削除するとデータが復活できなくなるため

すべてはDBへの操作

- PHPなどのプログラムはデータをどうやって取り出すか、加工するかということに主眼を置いて作成する
- データを保存する、ということはDBに任せる
- 条件をきちんと精査してデータを取り出す、更新する必要がある
- 取り出す条件を変更する場合は、ブラウザーのリフレッシュ（リロード）が必要
 - AjaxやPjaxを使えば不要になる

phpmyadmin



現在MySQL内にある
データベースの一覧

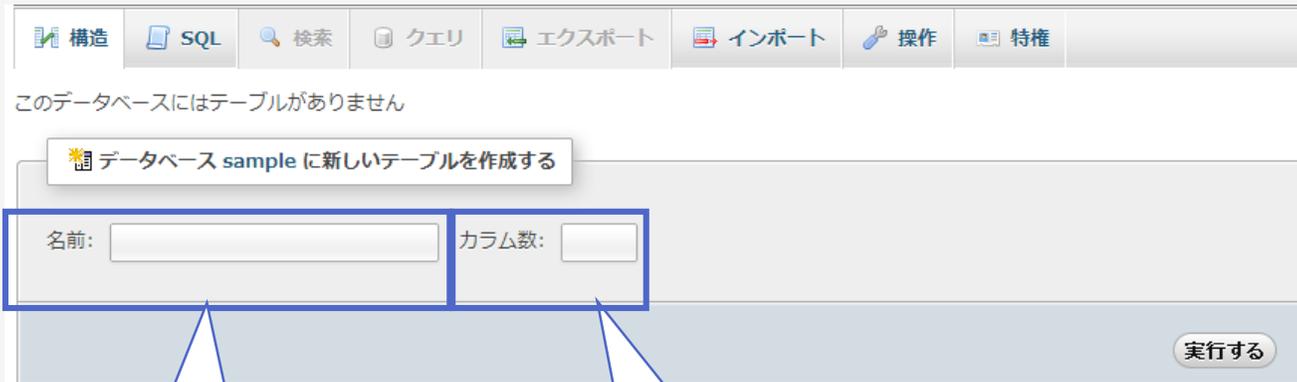
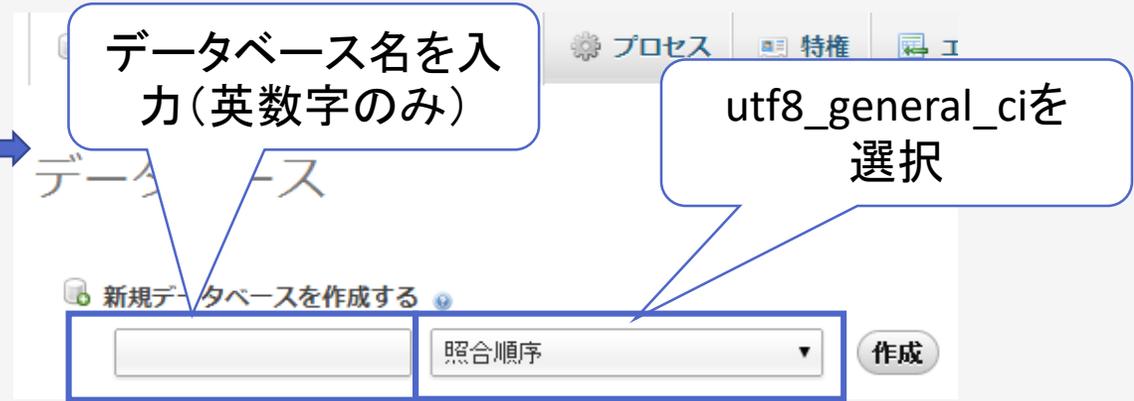
データベースをクリックすると
データベース内にある
テーブルの一覧が出てくる



操作関連は上のボタンを利用

表示をクリックすると
テーブル内に格納され
ているデータが表示され
る

phpmyadminによるデータベース作成



テーブル名を入力

必要な列数を入力

phpmyadminによるデータベース作成

テーブル名:
s1

カラム	種別	長さ/値1	デフォルト値2	照合順序	属性	ヌル(NULL)	インデックス
	INT		なし			<input type="checkbox"/>	---
	INT		なし			<input type="checkbox"/>	---
	INT		なし			<input type="checkbox"/>	---
	INT		なし			<input type="checkbox"/>	---

列名を入力

長さ、桁数を指定。
種類がTEXTなら指
定しない

何も指定しなくてOK

空の値を認めるな
らチェックする

列にどういうデータが入るかを指定
INT: 整数数字
VARCHAR: 可変長文字列
TEXT: 文字列
DATETIME: 日付

IDのように一意にするのであれば、
「A_」にチェックを入れておく
(データが入る度に1ずつ足して
いって一意のデータを作成してく
れる)

ほかと被らない一意であれば
PRIMARYかUNIQUEを指定
数字などではINDEXを指定
文字列には何も指定しないのが
一般的
PRIMARYは必ず1つは必要

Connect DB

How to connect DB

PDOで接続

- 現在のPHPではDBへの接続にはPDOという仕組みを利用する
 - 以前は違った
- PDOを利用することでDB自体のソフトが変更されても運用が可能である、という建前
 - 実際は無理だろうと思う
- PDOは便利な仕組みなところもあるが、取っつきづらいかも
 - PDO自体PHPのクラスとして作成されている

クラス

- オブジェクト指向言語で出てくる概念＝クラス
- クラス＝データと関数の集まり
- ある機能を実現するために関数群の集まりから作成するよりも1つのクラスという設計図に収めた方が使い回しが効くことから非常に良く使われる
 - 実体化することで実際に使用できる（クラス自体は設計図であるため）
 - 実体化＝インスタンス化
- 現在のプログラミング言語はほとんどがクラスを基本にしてプログラムを構築していく

PDOでDBに接続

```
try {  
    $pdo = new PDO('mysql:host=ホスト名;dbname=DB名;charset=utf8','ユーザー名','パスワード',array(PDO::ATTR_EMULATE_PREPARES => false));  
} catch (PDOException $e) {  
    exit('データベース接続失敗。'.$e->getMessage());  
}
```

- try文は指定された処理を実行してエラーが生じたらcatchに指定された部分を実行する
- newでインスタンスを作成
 - インスタンスを作成し、\$pdoに格納することでDBを利用できる

PDOで取り出し

```
$stmt = $pdo->query("SELECT * FROM テーブル名 ORDER BY no ASC");  
while($row = $stmt -> fetch(PDO::FETCH_ASSOC)) {  
    $title = $row["title"];  
    print($title);  
}
```

- PDOのqueryメソッドを利用してSQL文を発行する
- 結果は\$stmtに入ってくる
 - これをfetchメソッドを使って中身を取り出す
 - DBの結果がいくつ入っているか分からないため、ループで処理を行う

PDOでデータを挿入

```
$name = 'one';  
$value = 12;  
$sql = "INSERT INTO テーブル名 (name, value) VALUES (?,?)";  
$stmt = $pdo -> prepare($sql);  
$stmt->execute(array($name,$value));
```

- ユーザーから受け取ったデータはそのままDBに入れると危険
 - SQLインジェクション脆弱性
- prepareメソッドを利用し、一時的に別の文字（ここでは「?」）を指定しておく
 - 最後にexecuteメソッドを実行するときに変数を与えることで、問題の無いSQL文にすることが可能

PDOその他

```
$stmt = $pdo -> query("SELECT * FROM テーブル名");  
$count = $stmt -> rowCount();
```

- テーブル内にあるデータ数を数える
 - データが無かったら、、という処理をするときに利用