Webデザイン実習3B

2019/9/13

Kazuma Sekiguchi
class@cieds.jp

フォーム

- HTMLで作成されるユーザが入力できる唯一の画面
- 主要なタグ
  - <form>：フォームの開始を示す
  - <input>：ほとんどのフォーム部品
  - <textarea>：長いテキストを入力して貰うときの部品
  - <label>：部品の説明（ex：氏名など）

- ほとんどはinputタグのtype属性で変化させる
- name属性に固有の値を設定しておく
- 送信先はformのaction属性で指定する

最小限のフォーム

```
<form action=“mail.php” name=“form1” method=“post”>
<input type=“text” name=“name”>
<input type=“submit”>
</form>
```

- formのmethod属性：postかgetを選択。通常はpost
- action属性：データの送信先を指定
- input type=“text”：テキストフィールドの作成
- input type=“submit”：送信ボタンの作成
  - クリックすることで、テキストフィールドに入力されたデータが送信される

フォームの注意

- 必須などのチェックを行う機能は無い
  - HTML5から必須チェックを行うことが可能（ブラウザ依存）
  - JSでチェックを行う
- 正しいデータが入っているかは調べられない
  - 形式に沿っているかどうかはJSで判別可能
    - 電話番号、メールアドレスなど
- 送信ボタンを押すと、データの送信先に遷移する
  - 確認画面などはデータの送信先のプログラム上で作成する必要あり
  - JSを利用して、データだけを送ることも可能（遷移無し）

フォームの部品

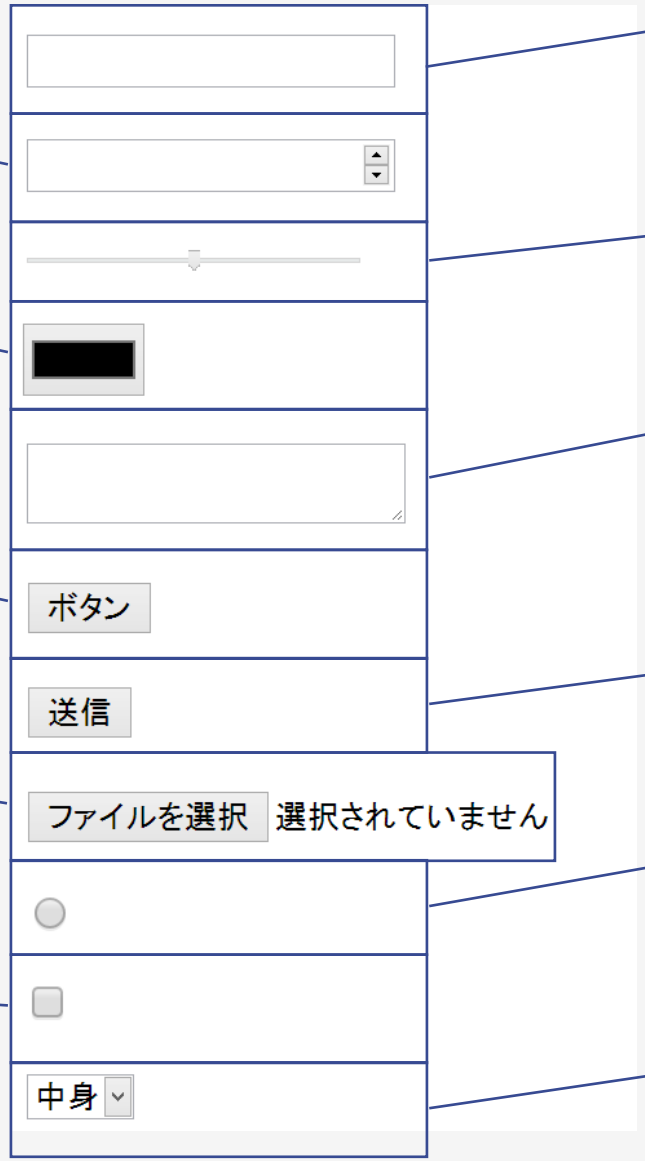
数値フィールド(HTML5)
<input type="number">

色選択(HTML5)
<input type="color">

ボタン
<input type="button">

ファイル
<input type="file">
保存されているファイルを
アップロードする場合に使用

チェックボックス
<input type="checkbox">



テキストフィールド
<input type="text">

範囲指定(HTML5)
<input type="range">

テキストエリア
<textarea>

送信ボタン
<input type="submit">

ラジオボタン
<input type="radio">
排他処理

セレクトボタン
<select>
中身は<option>で指定

フォームとCSS

- フォームの部品は全てCSSを適用可能
  - 色や枠線などもCSSで規定されているため、指定することで上書き可能
  - フォーム部品自体はインライン要素

- CSSで記述する場合は、属性値を含めた指定をすると指定しやすい

```
input[type=“text”]{
    background-color:#FFF;
}
```

テキストフィールドのみに適用

```
input[type=“submit”]{
    background-color:#FFF;
}
```

送信ボタンのみに適用

フォームのチェック

- ユーザに入力して貰う場面は意外に多い
  - ECサイトでの買い物

- 正しいデータが必要
  - 買って貰ったのに送れない・・・

- 入力を間違えると処理するプログラムのせいで、入力したデータが全て消えることがある（昔は良くあった）
  - ユーザの負担増

フォームのチェック（2）

- JavaScriptで予め入力データをチェック
  - 不足しているデータや間違えを指摘
  - ほぼリアルタイムで判別出来るので、ユーザの負担が減る
  - 入力したデータが全て消えることがない

- JavaScriptでチェックしてもサーバサイドで再度のチェックは必要
  - JavaScriptはユーザが任意にOffにできる
  - 受け取ったデータは必ずPHP側でもチェックする
  - HTML5のフォームチェックも回避が可能

正規表現（1）

- ユーザから入力してもらうデータは大体決まっている
  - メールアドレス、電話番号・・・などなど
- それぞれ決まり（パターン）がある
  - 電話番号：ハイフンと数字のみ
  - メールアドレス：英数字と一部の記号「@」必須
    @の直前に.が付かない

- このパターンが正しいかどうかチェックをすれば良い

正規表現（2）

- パターンをコンピュータに判別させるときの書式
- 英数字と記号を組み合わせてパターンを表現する
- ひらがなやカタカナも判別できるが、漢字は不可能（かなりトリッキーなことを使えば可能）

- テキストエディタなどで「検索」「置換」を行う際にも使える（便利！）

正規表現(3) PHPの場合

- 正規表現で入力されたデータが正しいか判断する場合
- preg_match関数を使用して引数に正規表現を記述
  - ereg関数は非推奨になっているので注意
- 正規表現は「/（スラッシュ）」の間に記述

```
    if(preg_match('/^[a-zA-Z0-9]+[a-zA-Z0-9¥._-]*[a-zA-Z0-9]+@[a-zA-Z0-9_-]+¥.[a-zA-Z0-9¥._-]+$/',$mail)){
        $str = '正しいメールアドレス';
    }
```

よく使う正規表現（1）

- ふりがな
  - ひらがなで書かれていることを検索

```
^[あ-ん]+$
```

- フリガナ
  - カタカナで書かれていることを検索
  - 半角カタカナを使わせないように全角カタカナだけを検索する

```
^[ァ-ヾ]+$
```

小文字のア

踊り字（おどりじで変換）

よく使う正規表現（2）

- 電話番号、FAX番号、郵便番号
  - 数字とハイフンだけで書かれていることを検索
  - 電話番号は桁数が違う（携帯は11桁など）ことに注意

```
^[0-9-]+$
```

よく使う正規表現（3）

- メールアドレス
  - RFC 2822(http://tools.ietf.org/htmlrfc2822)
  - @があること。@の直前に「.」がないこと
  - 「.」が連続しないこと
    - DocomoとAuでは許可している
  - 英数字と一部の記号で構成されていること（今後変わるかも）

```
^[a-zA-Z0-9._-]+[a-zA-Z0-9_]*@[a-zA-Z0-9._-]+¥. [a-zA-Z0-9._-]+$
```

ちなみに

- 厳密にメールアドレスを正規表現で表すと

```
/^(?!(?:(?:¥x22?¥x5C[¥x00-¥x7E]¥x22?)|(?:¥x22?[^¥x5C¥x22]¥x22?)){255,})(?!(?:(?:¥x22?¥x5C[¥x00-¥x7E]¥x22?)|(?:¥x22?[^¥x5C¥x22]¥x22?)){65,}@)(?:(?:[¥x21¥x23-¥x27¥x2A¥x2B¥x2D¥x2F-¥x39¥x3D¥x3F¥x5E-¥x7E]+)|(?:¥x22(?:[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x21¥x23-¥x5B¥x5D-¥x7F]|(?:¥x5C[¥x00-¥x7F]))*¥x22))(?:¥.(?:(?:[¥x21¥x23-¥x27¥x2A¥x2B¥x2D¥x2F-¥x39¥x3D¥x3F¥x5E-¥x7E]+)|(?:¥x22(?:[¥x01-¥x08¥x0B¥x0C¥x0E-¥x1F¥x21¥x23-¥x5B¥x5D-¥x7F]|(?:¥x5C[¥x00-¥x7F]))*¥x22)))*@(?:(?:(?!.*[^.]{64,})(?:(?:(?:xn--)?[a-z0-9]+(?:-[a-z0-9]+)*¥.){1,126}){1,}(?:(?:[a-z][a-z0-9]*)|(?:(?:xn--)[a-z0-9]+))(?:-[a-z0-9]+)*)|(?:¥[(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){7})|(?:(?!(?:.*[a-f0-9][:¥]]){7,})(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,5})?::(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,5})?)))|(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){5}:)|(?:(?!(?:.*[a-f0-9]:){5,})(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,3})?::(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,3}:)?)))?(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-9]{2})|(?:[1-9]?[0-9]))(?:¥.(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-9]{2})|(?:[1-9]?[0-9]))){3}))¥])$/iD
```

だそうです

（http://emailregex.com/より）

よく使う正規表現（4）

- URL
  - RFC 3305及びその他で規定
  （http://tools.ietf.org/html/rfc3305）
  - 通常ユーザに入力させるのはHTTP
  - 英数字及び記号から構成される（今後日本語も）

```
^https?:¥/¥/[-_.!~*¥'()a-zA-Z0-9;¥/?:¥@&=+¥$,%#]+$
```

リダイレクト

- データを受け取った後で処理を行い、処理が完了したら、別のページに飛ばすことがある

- PHPのheader関数を利用することで任意のページに飛ばすことが可能
  - ただし、何か表示するような命令を記述した後でheader関数を呼び出してもエラーになるので注意

header関数

- 任意のHTTPヘッダを送出する関数
  - HTTPヘッダにブラウザーを別のページへと遷移させるためのものがある
  - header関数を利用することで、任意のページへと飛ばすことが可能
  - 遷移させたいページは絶対パス（URL）で記述するのが正しいが、相対パスでも遷移はする

```
<?php
header('Location:絶対パス');
?>
```

メール送信

- PHP内部にあるmail関数を利用
  - 添付ファイルなどをしたい場合は、PHPMailerなどの外部ライブラリを利用するのが一般的
    - 外部ライブラリを使うことで、同じ処理をまた書かなくて良い
    - 多量のテストがされているため、バグが発生しない

メール送信

- 単なるテキストメールであれば、以下のコードで送信可能
  - マルチバイト関数が利用できることが条件
  - マルチバイト関数が利用できないと文字化けが生じる
- メール内での改行は「”￥n”」で記述する
  - コード内で改行しても意味が無い。また「’」では改行されない

```
mb_language("japanese");
mb_internal_encoding("utf-8");
$to = $email;
$subject = "フォームからのメール";//メールのタイトル
$body = 'フォームからのメール'."¥n".'名前：'.$name."¥n".'性別：'.$gender."¥n".'メールアドレス
'.$email."¥n".'内容'.$contents;
$from = "from@example.com";//メール差出人アドレス
mb_send_mail($to , $subject , $body , "From:".$from);//メールを送信
```

テキストファイルの保存

- テキスト情報を保存する場合
  1. サニタイズ処理を行っておく
  2. 指定したファイルを開く
     - fopen()関数で開く。第一引数で開くファイル名、第二引数で開き方を指定可能
     - “a”を指定するとファイルが無ければ新規に作成し、追記モードで開かれる（一番楽）
  3. 書き込む内容を生成する
     - テキストの場合は、CSV書式がベターと思われる
     - 1つの書き込みに付き、1行を利用する

テキストファイルの保存

- テキスト情報を保存する場合
  4. 実際に書き込むには、fwrite()関数を利用
     - 第一引数にfopen()関数の返値を指定、第二引数に書き込む内容を指定する

- 読み出す場合
  - fopen()関数で、”r”を指定し、読み出しモードにする
  - ループを利用して、!feof($fp)に到達するまでループを利用して中身を取り出す

テキストファイルの保存

- 読み出す場合
  1. fopen()関数で、”r”を指定し、読み出しモードにする
  2. ループを利用して、!feof($fp)に到達するまでループを利用して中身を取り出す
  3. fgets()関数でファイルから1行取り出すことが可能。これを終わりまで繰り返せば、全て取得することが可能

ファイルアップロード

- 画像ファイルなどをフォームを通してアップロードしてもらえる

- セキュリティ上のリスクが高いため、使用には注意が必要
  - 実行ファイル（PHPファイルやPerlのファイルなど多種）をアップロードされる可能性があることに注意

ファイルのアップロード

- Webの場合、ファイルをアップロードしてもらうのは画像ファイルなどに止めておくべき
  - GIF，PNG，BMP，JPEGなどのみ有効にする
  - WindowsやmacOSで使われているファイル形式以外も有効なので、注意
  - たとえば、PHP、Perlの実行形式など

ファイルのアップロード

- 指定したサーバ上のフォルダにアップロード
  - PHP上では、一時フォルダから指定のフォルダにコピーという形を取る
  - アップロードされたファイルには別名を付ける
    - 万が一同じファイル名のファイルがあった際のことを想定
    - 日時秒を付与したり、ランダム関数を使って文字列を付与

ファイル形式のチェック

- アップロードされたファイルの形式チェックは必須
  - 実行ファイル形式をアップされていないかチェックする
  - 拡張子によるチェックが一般的だが、確実では無い
  - 画像ファイルの場合、PHPの関数による画像形式チェックを併用した方が良い

ファイル形式のチェック

- 画像の場合画像の形式をチェック
- exif_imagetype関数が便利

```
<?php
if(file_exists($file_pass) && $type = exif_imagetype($file_pass)){
    switch($type){
        case IMAGETYPE_GIF: //gifの場合
        echo "IMAGETYPE_GIF";
        break;
        case IMAGETYPE_JPEG:  //jpgの場合
        echo "IMAGETYPE_JPEG";
        break;
        case IMAGETYPE_PNG:  //pngの場合
        echo "IMAGETYPE_PNG";
        break;
        default:   //どれにも該当しない場合
        echo "gif、jpg、png以外の画像です";
    }
}else{
    echo "画像ファイルではありません（もしくはファイルが存在しません）";
}
?>
```

画像ファイルの保存

- ファイルがアップロードされたきた場合
  1. ファイルの形式をチェック
     - 拡張子で判断（gif ， jpeg ， pngなど）
     - それ以外は全て撥ねる。実行可能形式をアップロードされたらサーバが乗っ取られる
     - 画像の場合は、exif_imagetype()関数を利用して、画像形式を取得する方がよりベター（拡張子のチェックだけでは不安）
  2. サーバに一時的に保存されているので、一時的な名前を取得し、新しいファイル名を付ける（万が一同じファイル名があった場合に上書き処理になってしまうため）

画像ファイルの保存

- ファイルがアップロードされたきた場合
  3. ファイルを新しい名前に変更して一時保存フォルダから移動する
     - move_uploaded_file（）関数を利用
     - LinuxなどのUnix系サーバでは、ファイルパーミッションを書き込み可能にすること
  4. ファイル名は新しく付けたので、それを利用して<img>タグなどで呼び出せば、確認画面を作成可能
     - 画像をリサイズする場合は、GD関数を利用する
     - リサイズする場合は縦横比に注意

別ファイルにPHPプログラムを記述

- JSでは<script>タグを利用することで、複数のJSファイルを読み込むことが可能
- PHPなど（JS以外の大体のプログラム）ではプログラムの文中で他のプログラムファイルを読み込むことが可能
  - 機能ごとにファイルを分ける
  - 再利用しやすいようにしておく
  - 最近ではJSも別の仕組みを使うことで擬似的に利用可能
- includeすると読み込むファイルがそのまま展開される
  - 指定した順番に読み込まれる

include ‘読み込むファイル名’;//読み込めないと警告出すがそのまま継続して処理される
require ‘読み込むファイル名’;//読み込めないとエラーで終了
include_once ‘読み込むファイル名’;//1度だけ読み込む。同じファイルが2度読み込まれたら読み込まずに警告
require_once ‘読み込むファイル名’; //1度だけ読み込む。同じファイルが2度読み込まれたらエラーで終了

サーバサイドとクライアントサイド

さまざまなデータを保存しておく

DBサーバ

MySQL、PostgreSQL、SQLite、Oracle、SQL Serverなど

Webサーバ

Apache、IIS、nginxなど

PHP , Python , Ruby
Perl , JavaScript
C , Java

要求

HTMLや画像など

JavaScript

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捜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

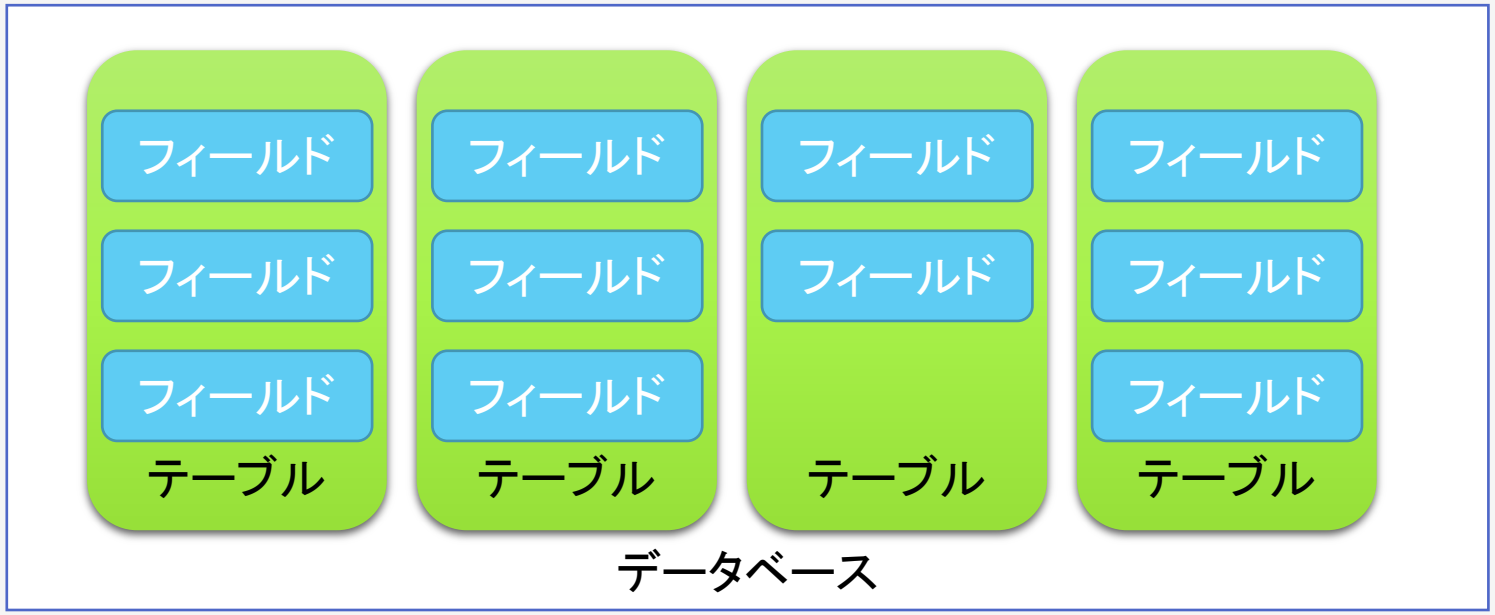
データベース（1）

- 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

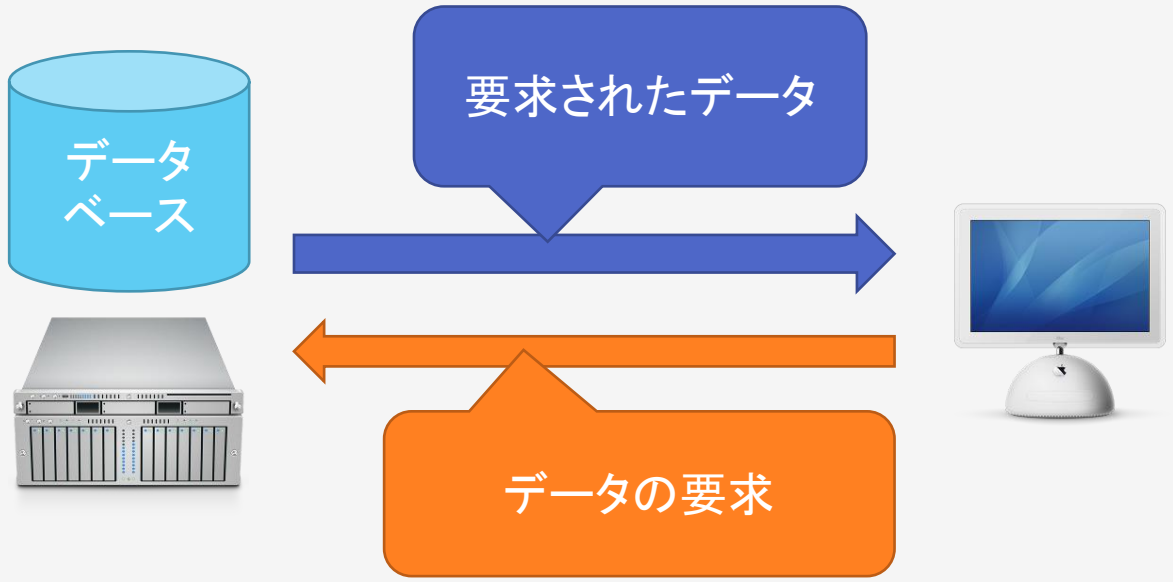
データベース（4）

- 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）

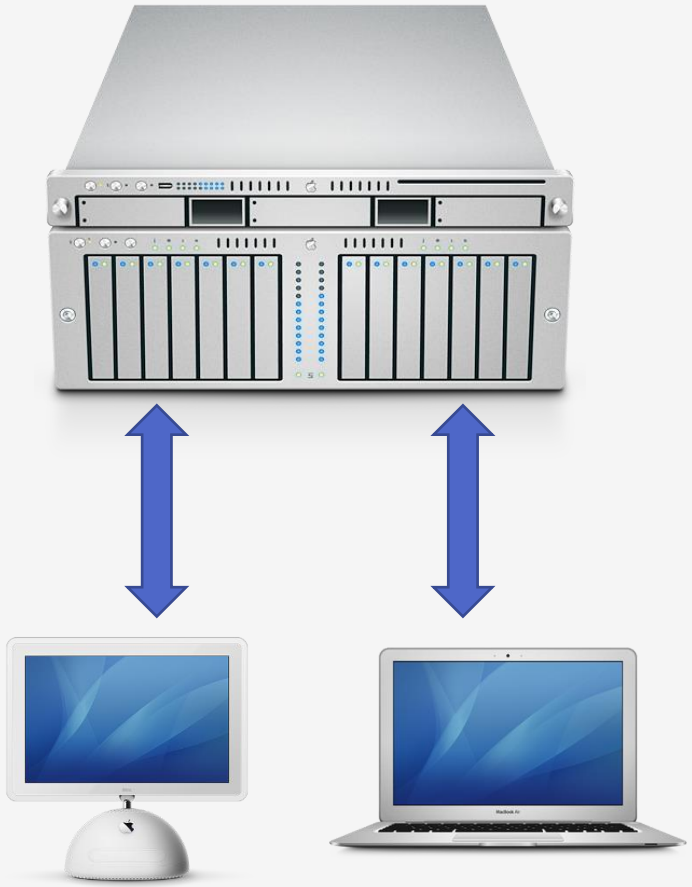
データベース（5）

- 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る
  - 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
  - SunMicrosystems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

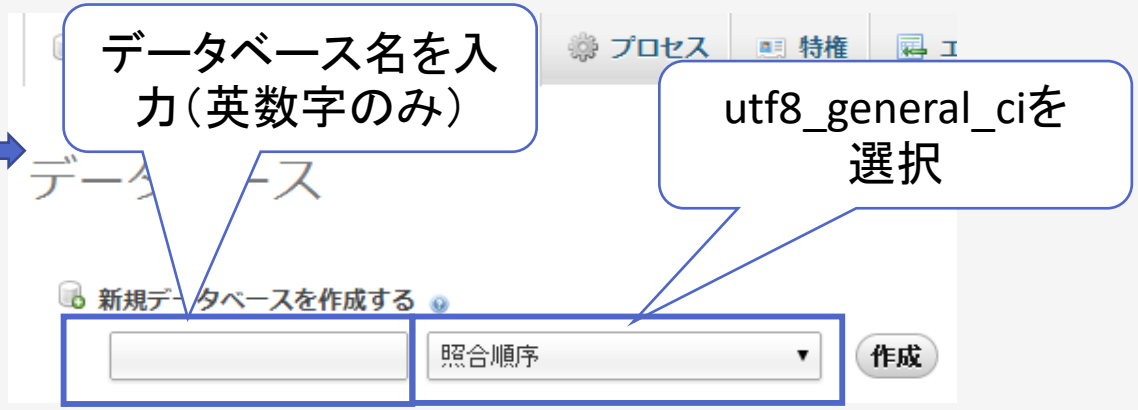
操作関連は上のボタンを利用

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

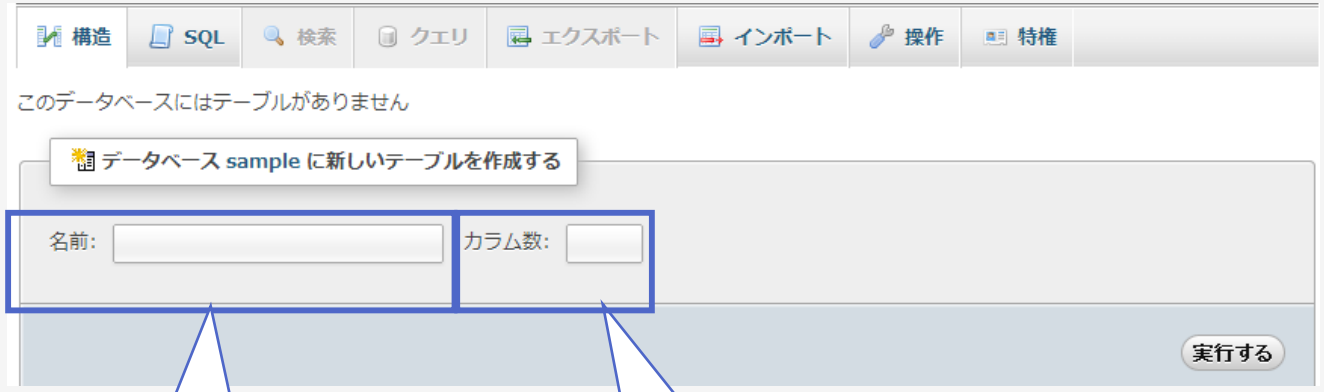
表示をクリックすると
テーブル内に格納されて
いるデータが表示される

phpmyadminによるデータベース作成



データベース名を入力（英数字のみ）

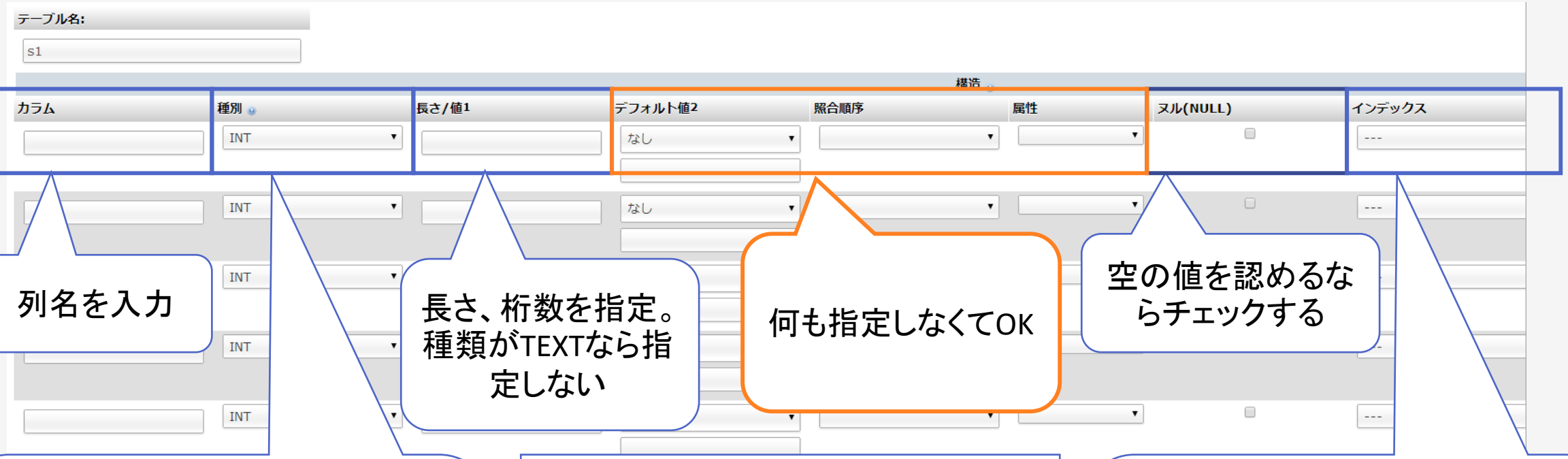
utf8_general_ciを選択



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



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_I」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UNIQE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