サーバサイドプログラミング

4. PDO (PHPからSQLアクセス)

コンテンツメディアプログラミング演習 Ⅱ

2014年

菊池，斉藤

1. PDO概要

- PDO (PHP Data Object)
  - PHP5.1から採用された汎用SQLの標準クラス.
  - オブジェクト指向を採用し，オブジェクトからメソッドやクラス変数を操作する.
  - MySQL, SQLiteなどのサーバソフトに依存せず，ほぼ共通のコードでプログラミングできる.

PHPからSQL文の実行

- PDOオブジェクト作成（準備）

```
$変数 = new PDO("ドライバ:
        host=ホスト;
        dbname=データベース名);
```

  - » ドライバ = mysql, sqlite など
  - » hostはローカルの場合は省略可能
  - » "dbname=" を指定しない（MySQLとの違い）
  - » $変数がメソッドなどを含むオブジェクト.

- ❑ 例）Mtsデータベースにアクセスする
  - » `$db = new PDO("sqlite:mts.sqlite");`
  - » `$db = new PDO("mysql:host=localhost;dbname=mts", "root", "パスワード");` （MySQLの場合）

SQLのエラーについて

- 注意
  - PDOのエラーは直接観測できない．事前にsqlite3コマンドで動作を確認しておくこと．
  - エラーを表示する設定（やらなくても動く）

```
$db = new PDO("sqlite:mts.sqlite");
$db->setAttribute(PDO::ATTR_ERRMODE,
     PDO::ERRMODE_WARNING);
```

SQLコマンドの実行

- query (クエリ)

  $変数->query(*SQL文*);

  » 任意のSQL文を実行する. $変数はPDOオブジェクト.

  ❑例)

  »
```
$db->query(
"INSERT INTO mts(name) VALUES('高尾山')"
);
```

  » この場合は返り値はなし(VOID)

検索結果の抽出 (1/3)

- fetch (物を取に行く)

```
$検索結果 = $変数->query(SELECT文);
$行 = $検索結果->fetch();
```

  - » SELECT文は複数の行と列を含むテーブルを返す
  - » fetch()はテーブルから1行づつ取り出す.
  - » 返し値 = False (もう行がない時)
    列から成る(通常の)配列
    列名をメンバとする連想配列

  - ❑ 例)
    - » `$rows = $db->query("SELECT * from mts");`
      テーブルmtsから全レコードを $rowsに取出す.
    - » $cols = $rows->fetch(); 1行づつ取り出す.

検索結果の抽出 (2/3)

- 例1） 配列で列を取出す
```
$cols = $rows->fetch()
print  $cols[0] . $cols[1] . $cols[2] . $cols[3].
"\n";
```
```
1谷川岳05
2丹沢06
```
- 例2)　配列を "-"で繋いで出力.
```
print  join($cols,"-") . "\n";
```
```
1-1-谷川岳-谷川岳-0-0-5-5-1227-1227
```
- 例3) 連想配列で名前と標高のみ出力.
```
print  $cols['name'] ."\t". $cols['height'] .
"\n";
```
```
1谷川岳05
2丹沢06
```
- 例4) 全ての行を出力.
```
while($cols = $rows->fetch()){
  print $cols['name'] . ",";
}
```
```
谷川岳,丹沢,天城山,八ヶ岳,那須岳,駒ヶ岳,燕岳,奥穂岳,
```

検索結果の抽出 (3/3)

- fetchAll (全ての行を抽出)

```
$検索結果 = $変数->query(SELECT文);
$表 = $検索結果->fetchAll();
```

  - » fetchAll()は1行づつではなく，全行を一度に表から取出す．
  - » 返し値 =「列名をメンバとする連想配列」の配列

❑ 例）

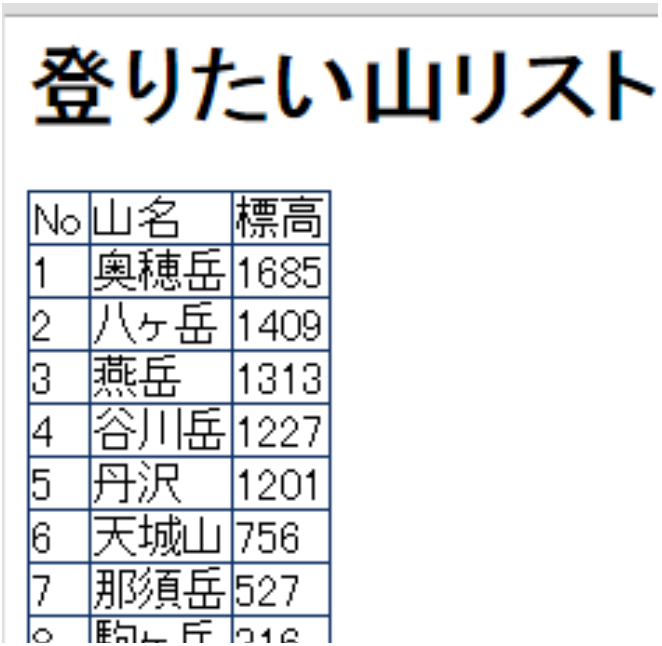
```
$rows = $db->query(
"SELECT * from mts");
$all = $rows->fetchAll();
print_r($all);
```

全行をまとめて表示

```
Array
(
    [0] => Array
        (
            [ID] => 1
            [0] => 1
            [name] =>谷川岳
             [1] =>谷川岳
             [day] => 0
            [2] => 0
            [hour] => 5
```

演習1

- Mts.sqlite データベースを用いて，標高順に山名を次の様に出力するmts-top.phpを書き，ブラウザから閲覧せよ.
  - ヒント: select でソートを行う. 必要な項目のみ選択. Tableタグ利用.
  - UTF-8のデータベース mts-u.sqlite (Webからダウンロード)



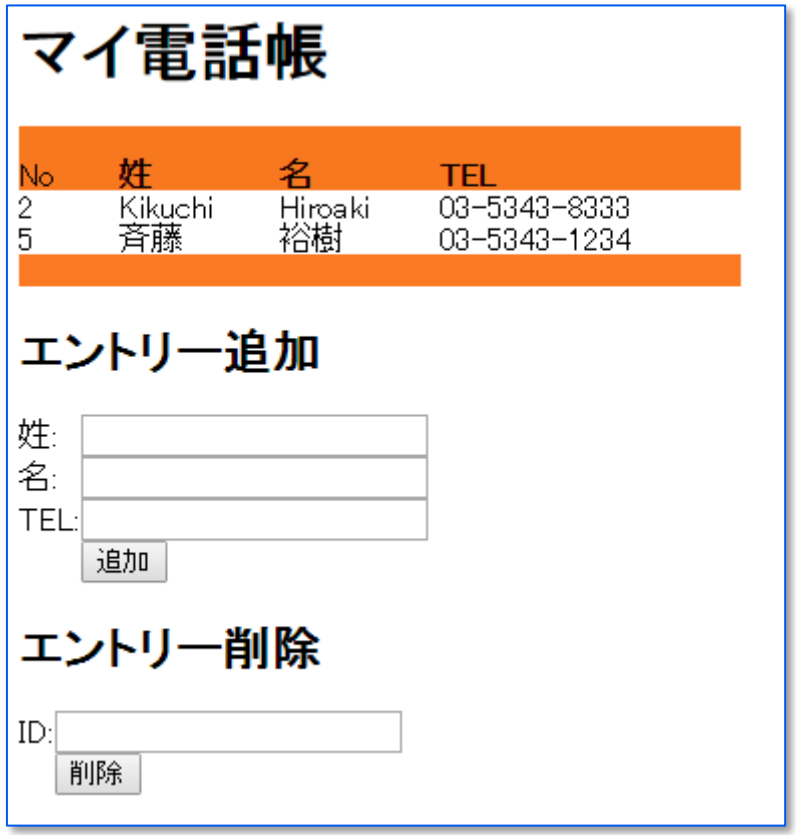
登りたい山リスト

No	山名	標高
1	奥穂岳	1685
2	八ヶ岳	1409
3	燕岳	1313
4	谷川岳	1227
5	丹沢	1201
6	天城山	756
7	那須岳	527

2. データベースの更新

- 「マイ電話帳」
  - 姓(lastname), 名(firstname), TEL(phone)の列から成る表 phonebook.sqlite

```
CREATE TABLE users
 (id integer primary key
autoincrement,
firstname text,
lastname text,
phone text)
```



(1) 全ての行を表示 list

- phonebook-list.php

```
<html><head>
<title> phonebook</title> <meta
charset="UTF-8"></head>
<body>
<h1> マイ電話帳 </h1>
<table border=0 cellpadding=0
cellspacing=0>
<tr bgcolor=#f87820>
<td width=50><br>No</td>
<td width=80><br><b>姓 </b></td>
<td width=80><br><b>名</b></td>
<td width=150><br><b>TEL</b></td>
<?php
$db = new
PDO("sqlite:phonebook.sqlite");
$result=$db->query("SELECT * FROM
users");
for($i = 0; $row=$result-
>fetch(); ++$i ){
        echo "<tr valign=center>";
        echo "<td >". $row['id'].
"</td>";
        echo "<td >".
$row['lastname']. "</td>";
        echo "<td >".
$row['firstname']. "</td>";
        echo "<td >".
$row['phone']. "</td></tr>";
        }
?>

</table>
</body></html>
```

(2) 行の追加

- queryメソッドでINSERT文実行

```
$db->query( "INSERT INTO テーブル
                VALUES（値）");
```

  - » $dbはデータベースを含むPDOオブジェクト （new PDO)
  - » 返し値はない. エラーのない様に値を用意.

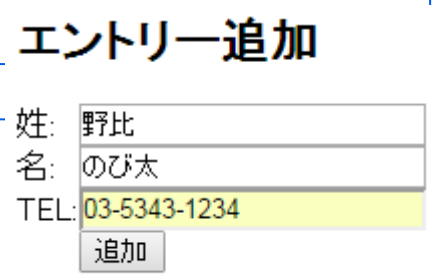
- ❑ 例）

```
» $firstname=$_GET['firstname'];
» $db = new PDO("sqlite:phonebook.sqlite");
»     $db->query("INSERT INTO users
  VALUES(null, '$firstname',null,null)");
```

FORMからのデータ受け取り

- phonebook-add.html

```
<html> <head><title> phonebook</title> <meta charset="UTF-8"> </head>
<body> <h2>エントリー追加</h2>
<form action=phonebook-add.php method=get>
<table border=0 cellpadding=0 cellspacing=0>
 <tr><td>姓:</td><td><input type=text size=20 name=lastname></td></tr>
 <tr><td>名:</td><td> <input type=text size=20 name=firstname></td></tr>
 <tr><td>TEL:</td><td> <input type=text size=20 name=phone></td></tr>
 <tr><td> </td><td><input type=submit border=0 value="追加"></td></tr>
</table>
</form> </body> </html>
```



- phonebook-add.php

```
<?php
$firstname=$_GET['firstname'];
$lastname=$_GET['lastname'];
$phone=$_GET['phone'];
$db = new PDO("sqlite:phonebook.sqlite");
$db->query("INSERT INTO users VALUES(null, '$firstname','$lastname','$phone')");
?>
Added.
```

$firstname PHPの変数
'firstname' Formのラベル
常に同じでなくてもよい

(3) 行の削除

- queryメソッドでDELETE文実行

```
$db->query( "DELETE FROM テーブル
                WHERE 条件");
```

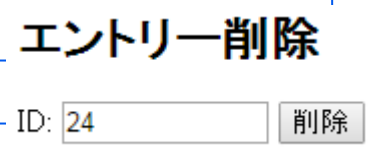
  - » 条件は，id=2 などの行を一意に決める式
  - » 返し値はない．

- ❑ 例）
  - » `$id=$_GET['id'];`
  - » `$db = new PDO("sqlite:phonebook.sqlite");`
  - » `$db->query("DELETE FROM users WHERE id='$id' ");`

行の削除

- phonebook-del.html

```
<html>
<head><title> phonebook </title> <meta charset="UTF-8"> </
head>
<body> <h2>エントリー削除</h2>
   <form action=phonebook-del.php method=get>
   ID: <input type=text size=10 name=id>
         <input type=submit value="削除">
   </form> </body>
</html>
```



- phonebook-del.php

```
<?php
$id=$_GET['id'];
$db = new PDO("sqlite:phonebook.sqlite");
$db->query("DELETE from users where id='$id'");
?>
deleted.
```

HtmlとPhpの統合

- phonebook-del2.php

```
<?php
if(isset($_GET['id'])) $id=$_GET['id'];
$db = new PDO("sqlite:phonebook.sqlite");
if(isset($id)) {
            $db->query("DELETE FROM users WHERE id='$id'");
}
?>
<html> <head> <title> phonebook </title> <meta charset="UTF-8">
</head> <body> <h2>エントリー削除</h2>
<form action=phonebook-del2.php method=get>
   ID: <input type=text size=10 name=id>
   <input type=submit value="削除">
</form>
<?php
$result=$db->query("SELECT * FROM users");
         print_r($result->fetchAll());
?>
</body> </html>
```

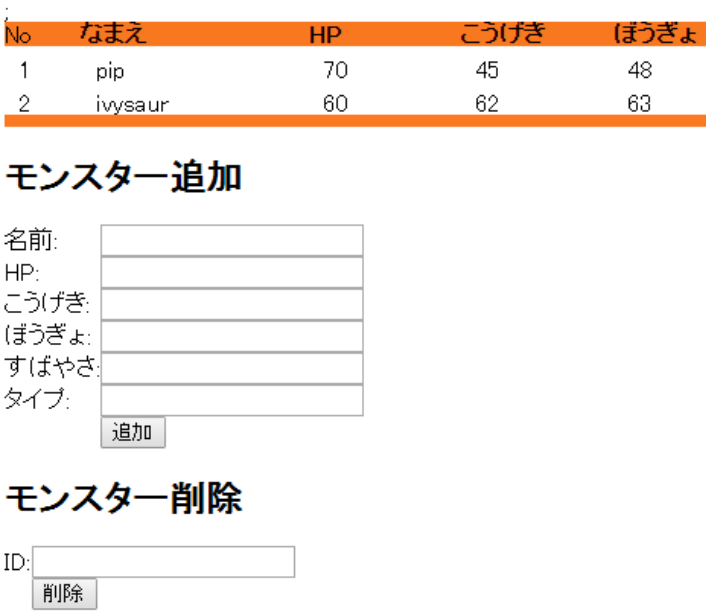
isset - 最初のアクセスの時にGET[id]が未定義で出るエラーを防止

全エントリーを表示して削除を確認する

演習2

- phonebook.phpを参考にして，ポケモン図鑑 pokemon.php を作れ.
  - pokemon.sqlite

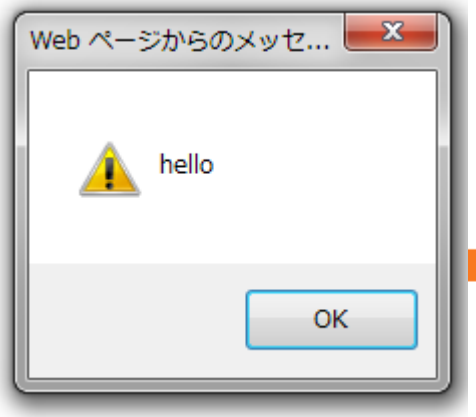
```
CREATE TABLE monsters (
id integer primary key
autoincrement,
name text,
hp integer,
offense integer,
defense integer,
speed integer,
type text);
```



  - 好きなポケモンを5匹追加せよ.
  - http://www.pokemon.co.jp/zukan/

3. セキュリティの考慮

- 演習2のモンスターの名前に次を入力して何が起きるか観察せよ.
  1. <h1> ピカチュー </h1>
  2. <script> alert('Hello') </script>
  3. <body bgcolor=black>

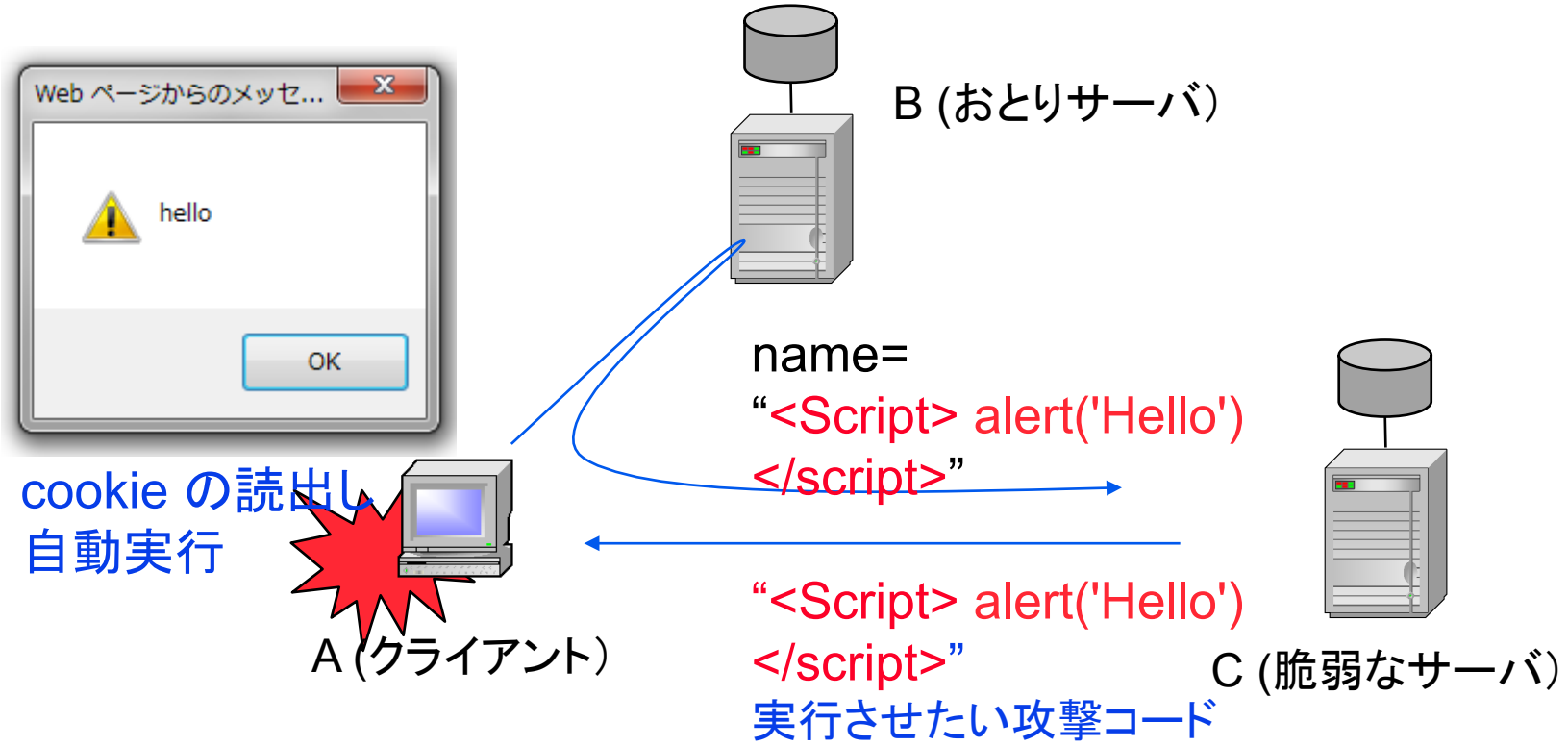


ポケットモンスターずかん

No	なまえ	HP	こうげき	ぼうぎょ		
1	pip	70	45	48		
2	ivysaur	60	62	63		
5	ピカチュー	60	30	60		
9	ピカチュー	10	10	10	10	10

クロスサイトスクリプティング

- Cross Site Scripting (XSS)



SQLインジェクション

- データベースへの攻撃
  - 意図しないSQL文の注入(injection)
  - 例）演習2のエントリーの削除（注意）

```
DELETE  FROM monsters WHERE
id = $id
```

  - $id＝「3 OR 1=1」

```
DELETE  FROM monsters WHERE
id = 3 OR 1=1 ;
```

  - 1=1は恒等式なので，常に成立し，全ての行を削除してしまう．(全データを抽出すると情報漏えい）

対策

- サニタイジング（衛生化）

  $変換後変数 = **htmlspecialchars**($入力変数);

  » 入力変数に含まれるタグをHTMLの実体参照に置き換える.

実体参照	&lt		&amp	&quot	&#9832
表記	<	>			

  - 例)

    入力:　　<h1>ピカチュー</h1>

    サニタイズ: <h1> ピカチュー </h1>

    （ブラウザでの見た目は同じ）

サニタイジングの準備

- 「htmlspecialchars」は長ったらしい！
  - そこで、自前で関数「h」を定義してしまおう
  - 各PHPプログラムの冒頭に以下の関数定義をして使う

```
function h($str) { return htmlspecialchars($str, ENT_QUOTES, "UTF-8"); }
```

  - ユーザから入力された情報はそのまま print/echo 文で表示するのではなく、必ず

```
print h($変数名);
```

のように関数「h」を介して表示する

その他の対策(参考)

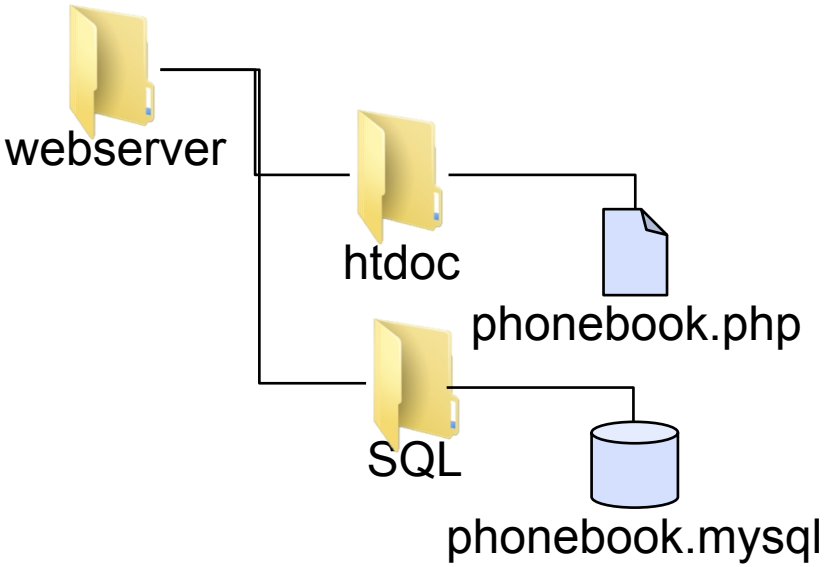
- サニタイジングだけでは完全にSQLインジェクションを防止できない.
  - タグや特殊記号を含まない命令
- 整数化
  - `$intid = round($id);` 文字列を整数のみに(切り捨て)
- 正規表現
  - `preg_match("/[0-9]+/", $入力変数)`
    入力が数字のみかどうかを判定する
- Prepared Statement
  - `$ps = $db->prepare("select * from tb where id=:A");`
  - `$ps->bindParam(":A", $a);`
  - `$ps->execute();`
    :Aの位置に$aの整数のみが代入.

データベースの漏えい

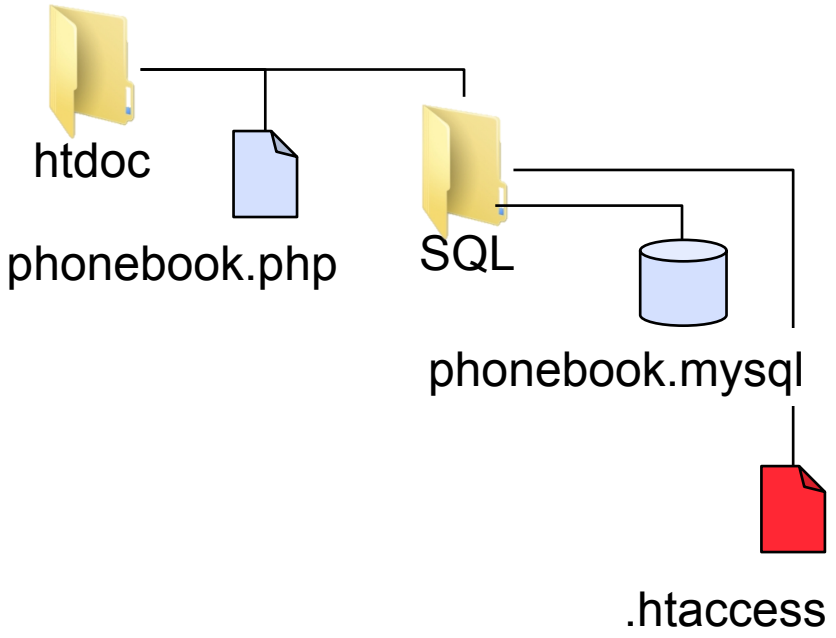
- ファイルとアクセスの関係
  - phpファイル （外部に公開する必要）
  - sqliteファイル （内部からのみアクセス）
- 攻撃
  - http://*localhost*/pokemon.sqlite でDBを丸ごとダウンロードする.（情報漏えい）

対策

- 1. SQL DBを外部のフォルダーに.



- 2. SQL　DBを直下の別フォルダーに.



- 欠点: ファイルの管理

アクセス制御ファイル

- .htaccess（ドットに注意）

```
Order deny,allow
Deny from all
```

  - deny（禁止），allow（許可）
  - order は，優先順序を指定.
  - このフォルダでは全てのウェブからのアクセスを禁じる（sqliteコマンドは別）

演習3

- 演習2で作成したpokemon.php を，次の2点を考慮した安全な pokemon-secure.php にせよ.
  - 検査方法

    (1) エントリー追加で名前「 <h1> ピカチュー </h1>」を持つ行を追加し，H1の大きさで表示されないこと.

    (2) データベース本体をhttp://localhost/pokemon.sqlite でブラウザからダウンロードできないことを確認.

演習4

- SQLの課題5 (オープンデータ)
  mydata.sqlite を用いて，それにレコード（行）を追加と削除する mydata.php を書け.
  - 追加する行のデータは任意に決めてよい.
  - 任意の機能を工夫した内容を，mydata-php.doc にて報告せよ.

提出物

- 提出用ドライブ (Y:)
  - コンテンツ..II (FMS)\第_回_組\2-組-番
  - 課題2. pokemon.php
  - 課題3. pokemon-secure.php, SQL \pokemon.sqlite, SQL\.htaccess
  - 課題4. mydata.sqlite, mydata.php, mydata-php.doc（報告書）

まとめ

- PDOはオブジェクト指向を用いた(　　)のSQLアクセスライブラリ．(　　)により任意のSQL文を実行する．
- PDOから行を追加するには，queryメソッドにより，SQLの(　　)を実行する．
- データベースを操作するコマンドにタグが入っていると(　　)の攻撃を受ける．防止するには，タグをHTMLの特殊文字に置き換える(　　)がある．