

HTML / PHP講習会

合宿事前講座 PHP編 Part.2(データベース) u306065 櫛田一樹

1: 前回やったこと

- PHPとは何か?
- PHPの基本的な文法とか……

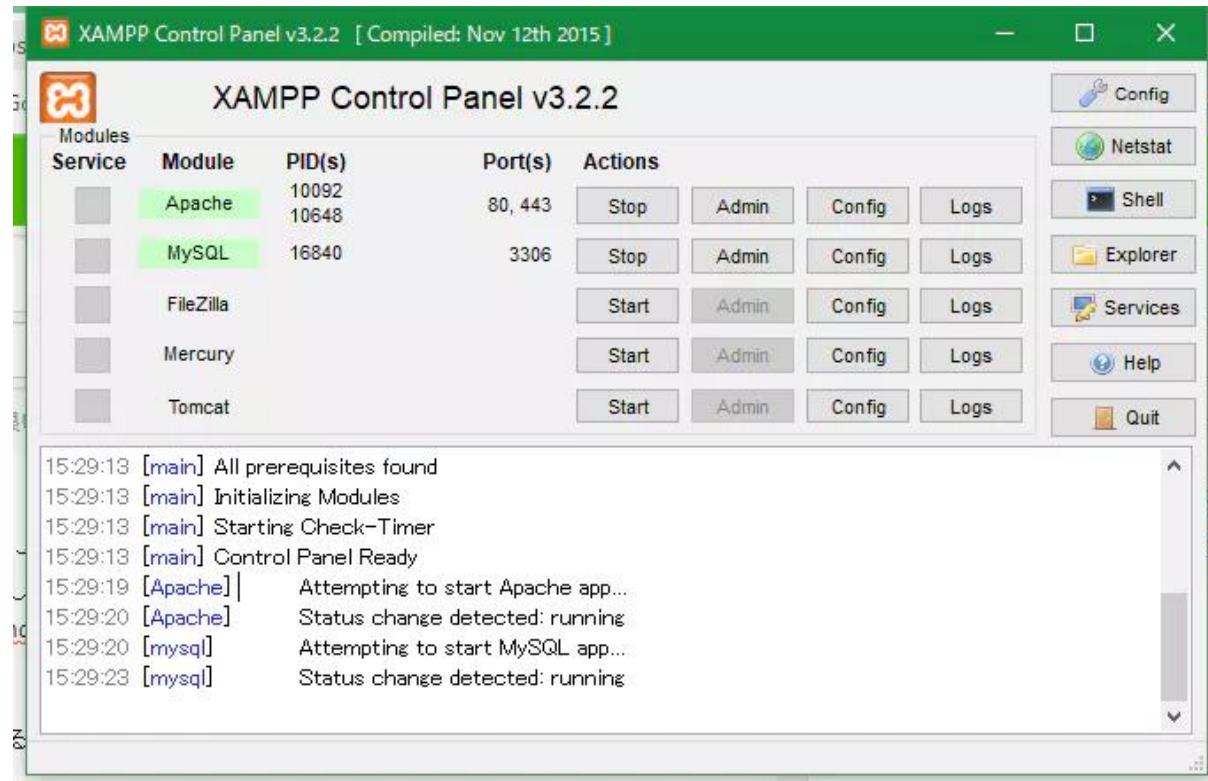
2: データベース

今回はデータベースを一緒に勉強していきましょう。

データベースを扱う上で最低限必要となる知識は前回ですべて学びました！
今回もがんばりましょう！

2-1: お勉強の前に

今回はデータベースを扱うので、MySQLというデータベースサーバーが必要になります。前回と同じようにApacheを立ち上げるのと同じく、MySQLも立ち上げましょう。



ApacheとMySQLを立ち上げて
左のような感じになつたらOK

3: Hello! DB!

おしながき

3-1: phpMyAdmin ! !

3-2: エクスポートとインポート

3-3: PHPからデータベースを操作しよう！

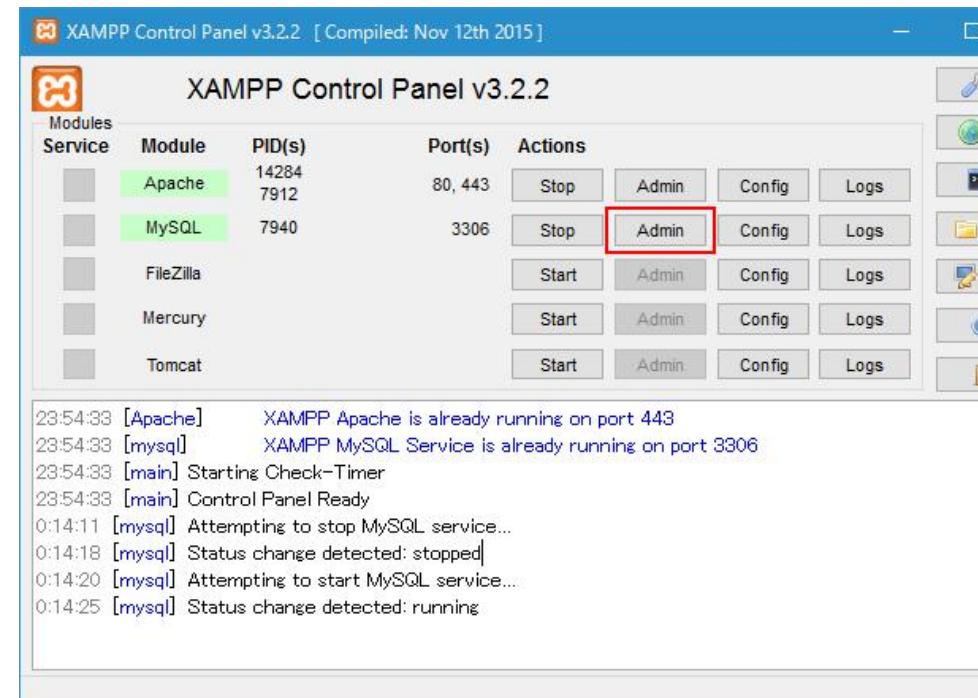
3-4: 取得したデータを表示しよう！

3-1: phpMyAdmin ! !

Hello! DB!

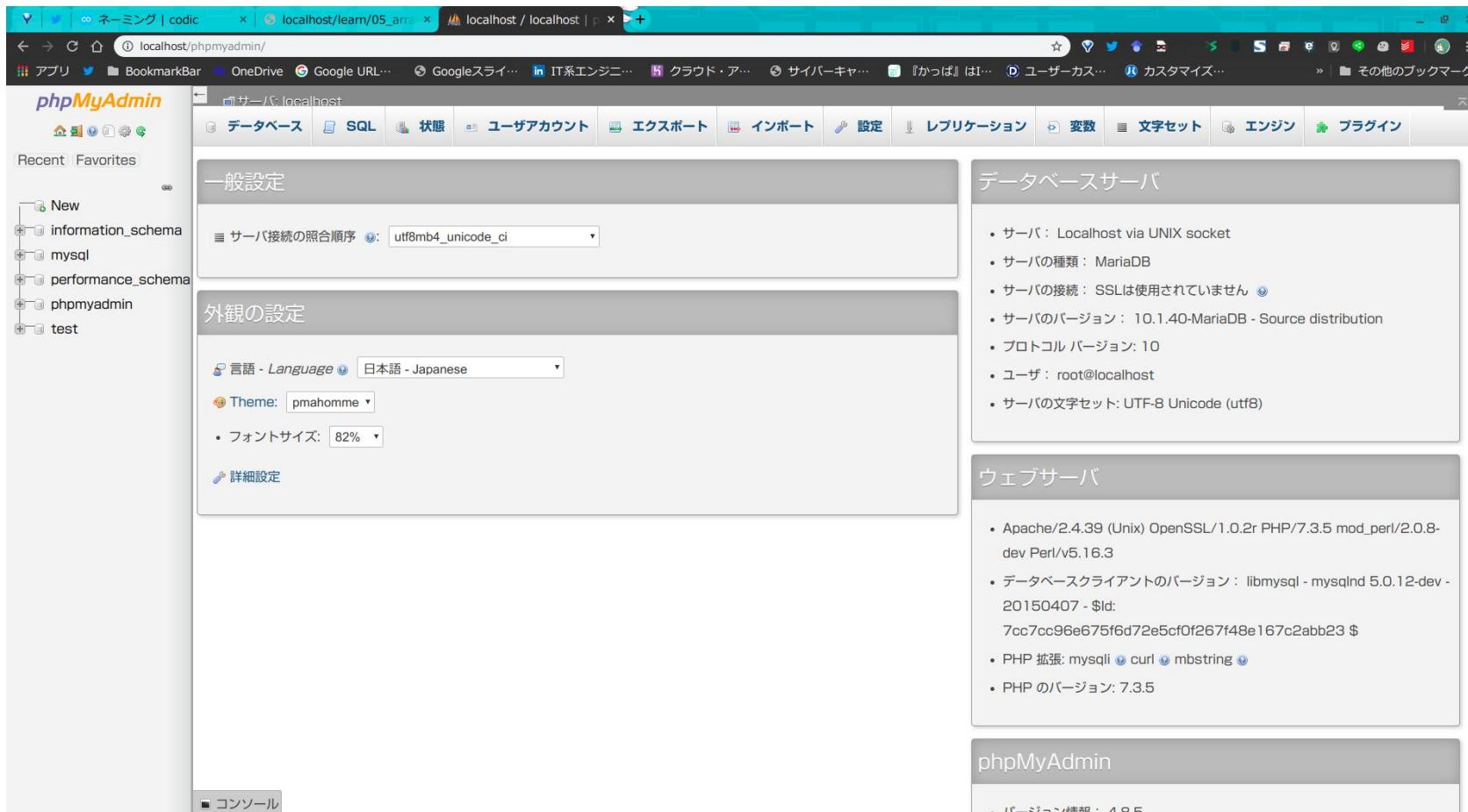
3-1: phpMyAdmin !!

さて、データベース編、始まりました。今回はデータベースだけが範囲なので結構ライトです。頑張っていきましょう！
まずは、XAMPPのMySQLのところの[Admin]をクリックしてください！



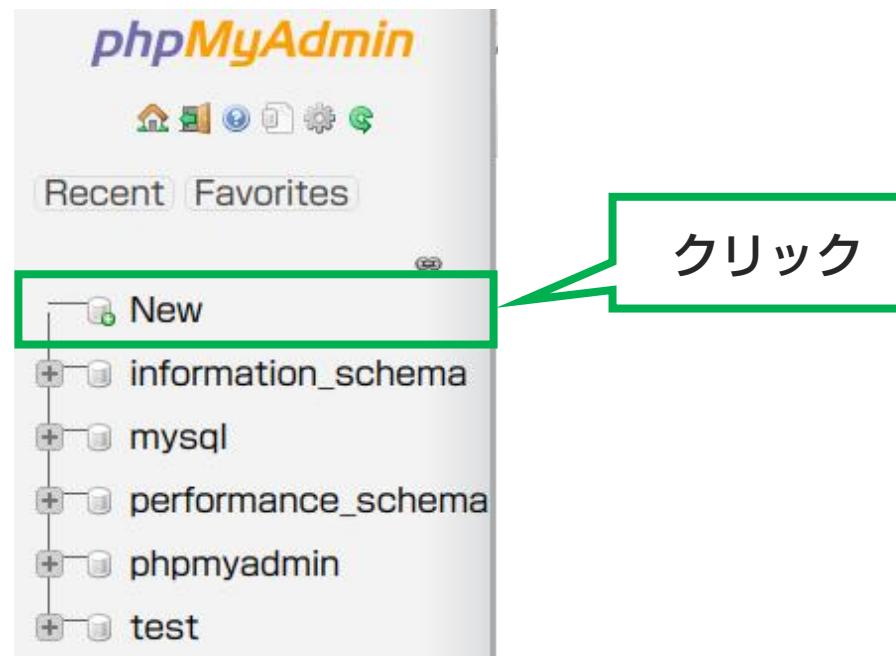
3-1: phpMyAdmin !!

すると、こんな感じの画面になると思います。これがphpMyAdminです。
ここから簡単にデータベースをいじれます！



3-1: phpMyAdmin !!

では、サイドバーのNewをクリックしてみましょう！



3-1: phpMyAdmin !!

データベース名をsampleとし、照合順序を utf8_general_ci としよう！



3-1-1: 照合順序って？？

文字列の検索に関する設定！！一般的には `utf8_general_ci` を設定するよ！

よく使う照合順序

- **`utf8_general_ci`**

→アルファベットの大文字, 小文字を区別しない

- **`utf8_unicode_ci`**

→大文字, 小文字 / 半角, 全角を区別しない

- **`utf8_bin`**

→完全一致

3-1: phpMyAdmin !!

下のようにテーブル作成画面が現れるよ！今回は試しにユーザーデータを管理する user テーブルを作つてみよう！



3-1: phpMyAdmin !!

名前をuser, カラム数を4に設定して実行ボタンをクリックしよう！



3-1: phpMyAdmin !!

すると、テーブルのカラムを設定する画面に遷移するので
以下のように設定しよう！設定の意味は後で詳しく解説します。

PRIMARY設定時にダイアログが現れるけどそのまま「実行」をクリック！

	名前	データ型	長さ	その他
カラム1	id	INT	10	インデックス： PRIMARY A_I : チェックを入れる
カラム2	name	VARCHAR	50	-
カラム3	age	INT	3	-
カラム4	email	VARCHER	100	NULL : チェックを入れる

3-1: phpMyAdmin !!

すべて記入すると以下のようになるから、「保存する」をクリックしよう！

構造

名前	データ型	長さ/値	デフォルト値	照合順序	属性	NULL インデックス	A.I.コメント
id	INT	10	なし			PRIMARY	PRIMARY
name	VARCHAR	50	なし			---	
age	INT	3	なし			---	
email	VARCHAR	100	なし			---	

3-1-2: データ型の種類

データベースでは保存時のデータ型を予め設定し、違う型のデータを入力しようとするとエラーになるよ！下の表でデータ型を確認しよう！

型	概要	例
INT	数値	-2147483648 ~ 2147483647
TINYINT	小さい数値	-128 ~ 127
VARCHAR	文字列	文字列の長さは65535バイトまで
TEXT	文字列	文字列の長さは65535バイトまで
DATETIME	日付	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
MEDIUMBLOB	バイナリ(2進数)ラージオブジェクト	画像などのデータを保存するときに使用する

3-1-2: データ型の種類

まずは INT VARCHAR DATETIMEを覚えよう！ varcharはPHPのstringに近いよ！ 電話番号に-(ハイフン)を加えた場合はvarcharになります。

型	概要	例
INT	数値	-2147483648 ~ 2147483647
TINYINT	小さい数値	-128 ~ 127
VARCHAR	文字列	文字列の長さは65535バイトまで
TEXT	文字列	文字列の長さは65535バイトまで
DATETIME	日付	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
MEDIUMBLOB	バイナリ(2進数)ラージオブジェクト	画像などのデータを保存するときに使用する

3-1-3: カラムの構成

長さとNULLとインデックスを設定しました。それぞれがどのような意味を持つか、下のクソ長文の書いてある表で確認しましょう。

長さ	長さは基本的にバイト長を表す。数字の場合型で上限下限が決まっているが、ここでは桁数で指定する。例えばINTで長さが「3」の場合、3桁までの整数のみ受け付ける。
NULL	NULL(空文字)を許すかどうかを設定する。チェックが無い場合、空のデータを書き込もうとするとエラーが発生する。
インデックス	同じ値を許さない場合、「PRIMARY」を選択する。PRIMARYを選択した場合、「A_I」にチェックを付ける場合が多い。A_Iとは、「オートインクリメント」の略称で、自動的に数字が割り振られる。

3-1: phpMyAdmin !!

それではいよいよデータの挿入と削除をやっていきましょう。
userテーブルを選択しているときに、ナビゲーションの「挿入」をクリック！



3-1: phpMyAdmin !!

すると、こんな画面が現れるよ！

The screenshot shows the 'Insert' page of phpMyAdmin. The top navigation bar indicates the server is localhost, the database is sample, and the table is user. Below the navigation is a toolbar with tabs: 表示 (View), 構造 (Structure), SQL, 検索 (Search), 插入 (Insert), エクスポート (Export), インポート (Import), 特権 (Privileges), 操作 (Operations), and SQL コマンド (SQL Command). The main area is titled 'Insert into user'. It displays four columns: id (int(10)), name (varchar(50)), age (int(3)), and email (varchar(100)). Each column has a dropdown menu for data type, a text input field for value, and a checkbox for NULL. A large text area at the bottom is labeled 'SQL' and contains the SQL query: 'INSERT INTO `user`(`id`, `name`, `age`, `email`) VALUES ('. A '実行' (Execute) button is located at the bottom right.

3-1: phpMyAdmin !!

では、次のように設定して実行ボタンを押そう！

カラム名	値
id	空(何も書かないで！)
name	自分の名前
age	自分の年齢
email	sample1@sample.com

3-1: phpMyAdmin !!

入力するとこんな感じになるよ！！そのまま実行ボタンをクリックしよう！

The screenshot shows the phpMyAdmin interface with the SQL tab selected. A new row is being inserted into a table with four columns: id, name, age, and email. The 'name' field contains '櫛田一樹', 'age' contains '20', and 'email' contains 'sample1@sample.com'. The 'id' field has a dropdown menu open. A checkbox is visible next to the 'email' input field. A red underline is under the 'sample1@sample.com' email address. A large '実行' (Execute) button is at the bottom right.

カラム	データ型	関数	NULL	値
id	int(10)			
name	varchar(50)			櫛田一樹
age	int(3)			20
email	varchar(100)		<input type="checkbox"/>	sample1@sample.com

実行

3-1: phpMyAdmin !!

完了文とその下に間にか文字列が出てきているね！

これは**SQL文**と言って

データベースを制御するときに必要になる文なんだ。

✓ 1 行挿入しました。

id 1 の行を挿入しました

```
INSERT INTO `user` (`id`, `name`, `age`, `email`) VALUES (NULL, '櫛田一樹', '20', 'sample1@sample.com');
```

3-1: phpMyAdmin !!

さて、今のSQL文を見てみよう！SQLはシンプルでわかりやすいねえ……

```
INSERT INTO `user` (`id`, `name`, `age`, `email`) VALUES (NULL, '櫛田一樹', '20', 'sample1@sample.com');
```

INSERTはデータを挿入する命令、INTOの後ろはテーブル名
()内にはデータを挿入したいカラム名を指定するよ！

VALUESに挿入したい内容を入れる！文字列にはシングルクオーテーション
をつけよう！ちなみにバッククオートは省略できるから下のように書けるよ！

```
INSERT INTO user ( id, name, age, email) VALUES (NULL, '櫛田一樹', '20', 'sample1@sample.com');
```

3-1: phpMyAdmin !!

では次はSQL文でデータの挿入を行ってみよう！SQLタブをクリックして下の画面を出そう！



このとき、デフォルトのSQL文は使用しないから削除してね！

3-1: phpMyAdmin !!

削除したら、下のように書こう！

INSERT INTO user (id, name, age, email) VALUES (NULL, '情研太郎', '18', 'sample2@sample.com');

入力したら実行ボタンをクリックしよう！

The screenshot shows the phpMyAdmin interface with the SQL tab selected. The main area contains the SQL query:

```
1 INSERT INTO user (id, name, age, email) VALUES (NULL, '情研太郎', '18', 'sample2@sample.com');
```

To the right, a column list shows the table structure:

カラム
id
name
age
email

Below the query, there are several buttons: SELECT*, SELECT, INSERT, UPDATE, DELETE, クリア (Clear), フォーマット (Format), and 自動保存されたクエリを取得 (Get saved query). There is also a checkbox for Bind parameters.

At the bottom, there is a bookmarking section and a toolbar with options for delimiter, query display, and execution.

3-1: phpMyAdmin !!

挿入に成功すると、先程と同じような画面が出てくるよ！

The screenshot shows the phpMyAdmin interface for the 'user' table in the 'sample' database. The top navigation bar includes links for '表示' (View), '構造' (Structure), 'SQL', '検索' (Search), '挿入' (Insert), 'エクスポート' (Export), 'インポート' (Import), '特権' (Privileges), '操作' (Operations), 'SQL コマンドの追跡' (SQL Command History), and 'トリガ' (Triggers). The '挿入' (Insert) button is highlighted. Below the navigation, there is a message box with a green checkmark indicating '1 行挿入しました。' (1 row inserted) and 'id 2 の行を挿入しました (Query took 0.0226 seconds.)'. The SQL query shown is: `INSERT INTO user (id, name, age, email) VALUES (NULL, '情研太郎', '18', 'sample2@sample.com')`.

3-1: phpMyAdmin !!

それでは、表示タブをクリックして、データが挿入されたことを確認しよう
うん！2件ともちゃんと格納されているね！成功だ！！



The screenshot shows the 'Display' tab of the phpMyAdmin interface. At the top, there's a toolbar with a search bar and several icons. Below it is a table header with columns labeled 'id', 'name', 'age', and 'email'. Two rows of data are listed below the header. Each row has a checkbox, edit, copy, delete, and export links. The first row contains the values: id=1, name=櫛田一樹, age=20, email=sample1@sample.com. The second row contains the values: id=2, name=情研太郎, age=18, email=sample2@sample.com.

	Display			
	Table structure for database: sample			
	Structure			
	id	name	age	email
<input type="checkbox"/>	編集	コピー	削除	1 櫛田一樹 20 sample1@sample.com
<input type="checkbox"/>	編集	コピー	削除	2 情研太郎 18 sample2@sample.com

[+ オプション](#)

[←](#) [→](#)

[Check all](#) チェックしたものを: [編集](#) [コピー](#) [削除](#) [エクスポート](#)

3-1: phpMyAdmin !!

では、この調子で以下SQL文で2つデータを入れてみよう！
先程と同じようにSQLタブを開こう！

複数の命令を一度に実行できるよ！SQLを2行入れてから実行を押してみよう！

```
INSERT INTO user (id, name, age, email) VALUES (NULL, '情研二郎', '17', 'sample3@sample.com');  
INSERT INTO user (id, name, age, email) VALUES (NULL, '日大花子', '49', 'sample4@sample.com');
```



3-1: phpMyAdmin !!

OK!!

+ オプション

	← ↑ →		id	name	age	email	
<input type="checkbox"/>		コピー	削除	1	櫛田一樹	20	sample1@sample.com
<input type="checkbox"/>		コピー	削除	2	情研太郎	18	sample2@sample.com
<input type="checkbox"/>		コピー	削除	3	情研二郎	17	sample3@sample.com
<input type="checkbox"/>		コピー	削除	4	日大花子	49	sample4@sample.com

サーバ: localhost » データベース: sample » テーブル: user

表示 構造 SQL 検索 挿入 エクスポート インポート

クエリボックスを表示

✓ 1 行挿入しました。
id 3 の行を挿入しました (Query took 0.0101 seconds.)

```
INSERT INTO user (id, name, age, email) VALUES (NULL, '情研二郎', '17', 'sample3@sample.com')
```

✓ 1 行挿入しました。
id 4 の行を挿入しました (Query took 0.0111 seconds.)

```
INSERT INTO user (id, name, age, email) VALUES (NULL, '日大花子', '49', 'sample4@sample.com')
```

3-1-4 MEMO: CRUD機能って？

Webサービスは基本的にCRUD(クラッド)機能からできています。
CRUDとは以下のそれぞれの頭文字をまとめたものです。

- C Create : 作る
SQL文では**INSERT**という命令文になる。
- R Read : 読む
SQL文では**SELECT**という命令文になる。
- U Update : 更新する
SQL文では**UPDATE**という命令文になる。
- D Delete : 削除する
SQL文では**DELETE**という命令文になる。

3-1-5: 色々なSQL文

CRUD機能だけで多くのWebサービスの機能が実現できます。
簡単なSQL文を確認しよう！

3-1-5-1: SELECT文

データベースから値を読み出す命令文。基本/応用情報技術者試験の常連。

SELECT カラム名 FROM テーブル名 WHERE 条件式;

実行してみよう！

```
SELECT id, name FROM user WHERE id = 2;
```

ちなみにSQLでの = はPHPとは異なり、等式を表します。

SELECT文にはこの他にも、取得するデータ数を制限するLIMITや
指定したカラムの値で行を並び替えるORDER BYなど様々な表現があるよ！

3-1-5-1: SELECT文

こんな感じの実行結果になれば大成功！！

✓ 行 0 - 0 の表示 (合計 1, Query took 0.0009 seconds.)

```
SELECT id,name FROM user WHERE id = 2
```

■ すべて表示 | 行数: 25 ▾ 行フィルタ: このテーブルを検索

+ オプション

id	name
2	情研太郎

SELECT id,name FROM user WHERE id = 2;

user表から idが2のデータのうち、idとnameを取り出しているね！

3-1-5-2: UPDATE文

SET以降に書かれたカラム名のデータを置き換える(更新する)命令だよ！

UPDATE テーブル名 SET カラム名 = 値 WHERE 条件式;

実行してみよう！

```
UPDATE user SET email = 'new_mail@new_mail.com' WHERE id = 1;
```

文字列にはシングルクオーテーションが必要なので忘れずに！！

3-1-5-2: UPDATE文

SET以降に書かれたカラム名のデータを置き換える(更新する)命令だよ！

UPDATE テーブル名 SET カラム名 = 値 WHERE 条件式;

実行してみよう！

```
UPDATE user SET email = 'new_mail@new_mail.com' WHERE id = 1;
```

文字列にはシングルクオーテーションが必要なので忘れずに！！

3-1-5-2: UPDATE文

以下のような表示になつたら見てみよう！

✓ 1 行変更しました。 (Query took 0.0061 seconds.)

UPDATE user SET email = 'new_mail@new_mail.com' WHERE id = 1

表示タブをクリックして、id = 1のmailを見てみよう！変更されてるね！

			id	name	age	email	
<input type="checkbox"/>	 編集	 コピー	 削除	1	櫛田一樹	20	new_mail@new_mail.com
<input type="checkbox"/>	 編集	 コピー	 削除	2	情研太郎	18	sample2@sample.com
<input type="checkbox"/>	 編集	 コピー	 削除	3	情研二郎	17	sample3@sample.com
<input type="checkbox"/>	 編集	 コピー	 削除	4	日大花子	49	sample4@sample.com

UPDATE user SET email = 'new_mail@new_mail.com' WHERE id = 1;

user表のidが1のemailのカラムのデータをnew_mail@new_mail.comに変更しているね！

3-1-5-3: DELETE文

WHERE句で指定したエリアのデータを削除するよ！

DELETE FROM テーブル名 WHERE 条件式;

実行してみよう！

`DELETE FROM user WHERE id = 1;`

3-1-5-3: DELETE文

以下のような表示になつたら見てみよう！

✓ 1 行変更しました。 (Query took 0.0122 seconds.)

DELETE FROM user WHERE id = 1

表示タブをクリックして、id = 1を見てみよう！きれいサッパリ消し飛んでるね！

+ オプション		←→	▼	id	name	age	email
<input type="checkbox"/>		編集		コピー		削除	2 情研太郎 18 sample2@sample.com
<input type="checkbox"/>		編集		コピー		削除	3 情研二郎 17 sample3@sample.com
<input type="checkbox"/>		編集		コピー		削除	4 日大花子 49 sample4@sample.com

DELETE FROM user WHERE id = 1;

user表のidが1のデータを削除しているね！

3-1-5: 色々なSQL文

以上が大事なSQL文だよ！
今後データベースを使ったアプリ開発で
いろいろ使うのでおぼえておくと幸せになれるよ！！

3-1-6 MEMO: SQLとセキュリティ

近年、何かとセキュリティが話題に挙がっていますが、DB, SQLでももちろん、実装によって様々なセキュリティの問題が出てきます。

有名なデータベース関連の攻撃手法といえば「**SQLインジェクション攻撃**」でしょうか(基本情報/応用情報では頻出です！！)。これは、Webサイトの入力フォームに不正なSQL文が入力された場合、入力されたSQL文によってはデータベースのすべてのデータが抜き取られてしまったり、消されてしまったりする、厄介な攻撃手法です。これもエスケープ処理やバリデーションで十分防ぐことのできる脅威です。まあ時間があるときにでも調べてみると面白いかもしれません。

今回はガチガチの合宿ではなく、割とゆるめの合宿なので、そこまで対策をする必要はありませんが、「このような攻撃もある」というところは頭の片隅にでも入れておくと将来役に立つかかもしれませんね！もちろんSQLインジェクション攻撃以外にも様々な攻撃手法があるので、気になった方は調べて見ると面白いかもしれません。セキュリティはいいぞ！！

3-2: エクスポートとインポート

Hello! DB!

3-2: エクスポートとインポート

先日、AWSの東京リージョンで大規模通信障害が発生しましたね。
ガルパとか、アズレンとか、駅メモとか、AtCoderとか……

様々なゲームが一時的に遊べなくなったりらしいです。

3-2: エクスポートとインポート

えっと、ここで言いたいのは
「この先何が起こるかわからない！」
ということです。

今回のようにサーバに繋げなくなってしまったり、
はたまたデータが全部消えて無くなってしまったり……！

3-2: エクスポートとインポート

まあ、別に合宿だし、**開発以上に楽しむことを優先していただきたい**ので
そこまでうるさく言うつもりはないですが…

とりあえず将来データベースいじる方も、そうでない方も。

備えあれば憂いなしです！

phpMyAdminでもデータベースのバックアップが取れるので
一緒に取っていきましょう！

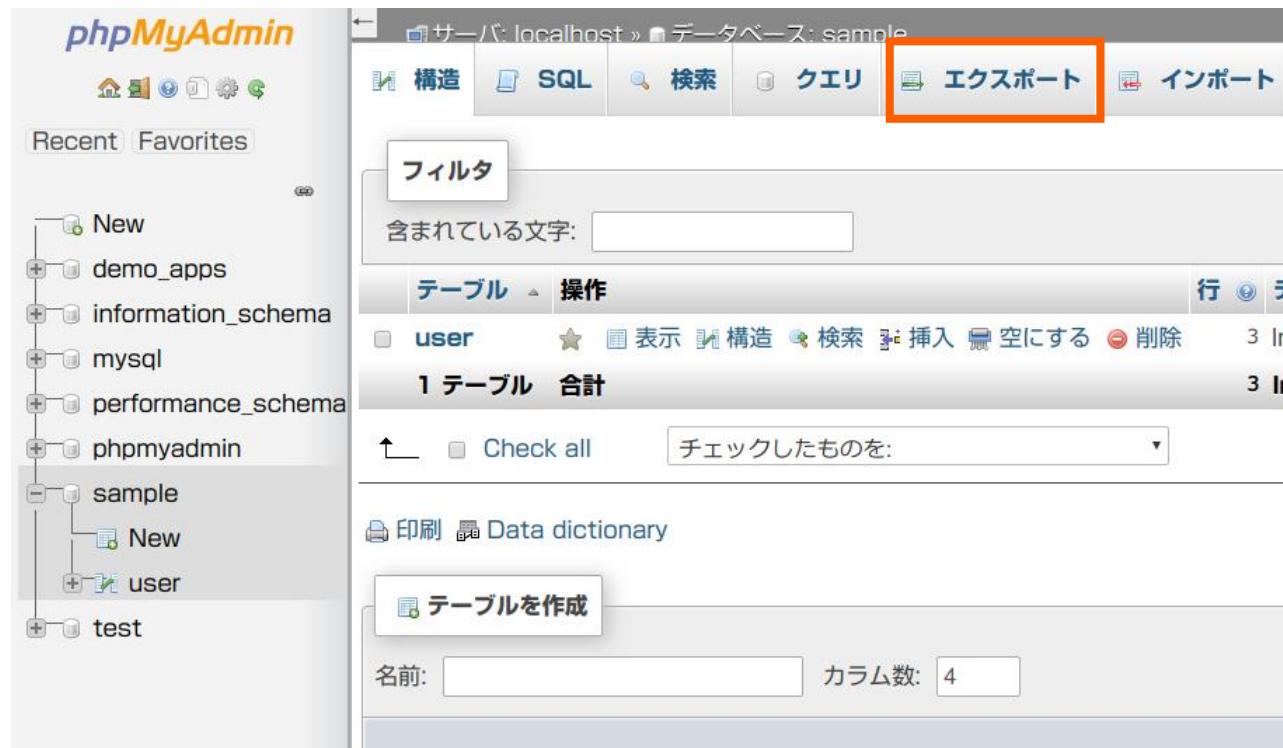
3-2: エクスポートとインポート

このセクションではそこまで詳しくはやりません。
やり方だけです。

サラッと聞き流していただいて構わないです。

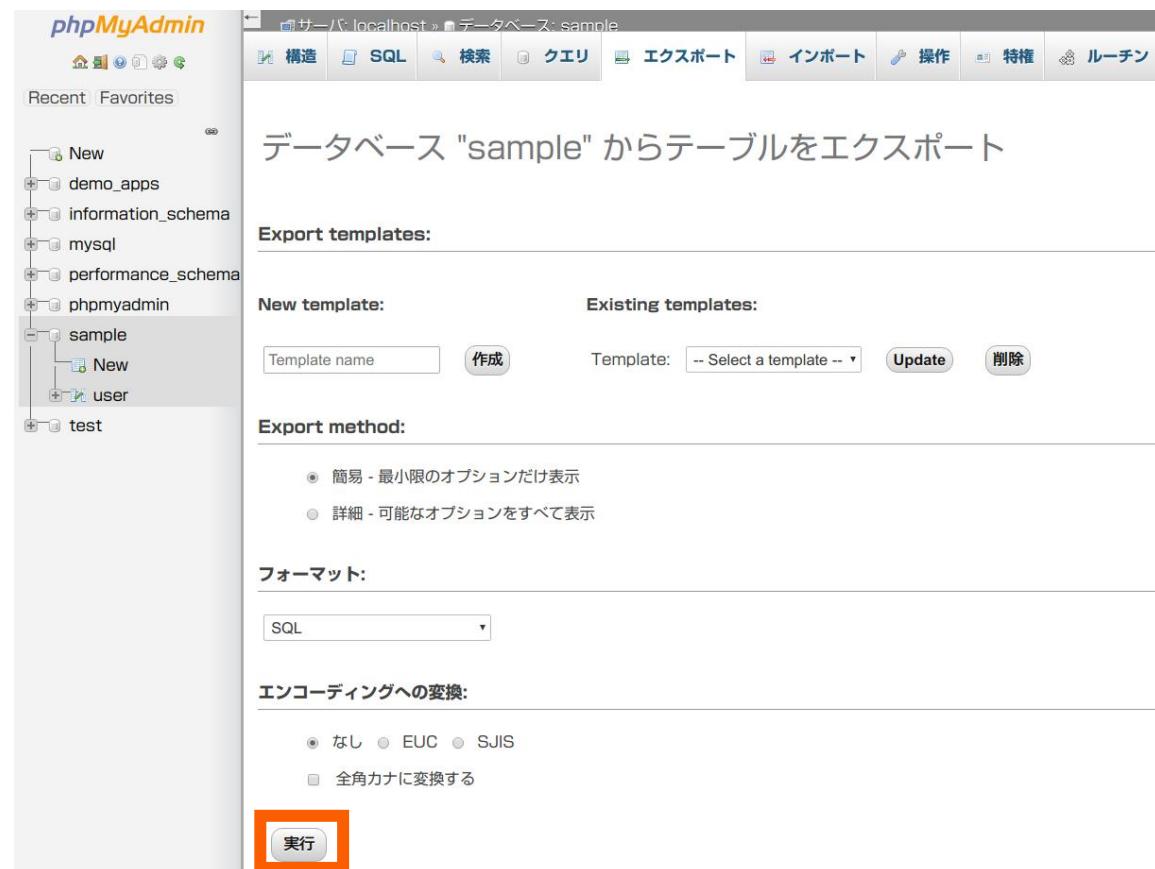
3-2: エクスポートとインポート

先程作ったsampleデータベースを選択した状態で、
ナビゲーションの「エクスポート」を押してみましょう。



3-2: エクスポートとインポート

こんな感じになつたらそのまま実行をクリック！
sqlファイルがダウンロードされます。



3-2: エクスポートとインポート

ダウンロードされたsqlファイルを見てみると、
何やらSQL文が書かれています。



The screenshot shows a code editor window with the file 'sample.sql' open. The file path is 'learn > 06_database > download > sample.sql'. The code itself is as follows:

```
sample.sql
learn > 06_database > download > sample.sql
27 --
28 -- テーブルの構造 `user`
29 --
30
31 CREATE TABLE `user` (
32     `id` int(10) NOT NULL,
33     `name` varchar(50) NOT NULL,
34     `age` int(3) NOT NULL,
35     `email` varchar(100) DEFAULT NULL
36 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
37
38 --
39 -- テーブルのデータのダンプ `user`
40 --
41
42 INSERT INTO `user` (`id`, `name`, `age`, `email`) VALUES
43 (2, '情研太郎', 18, 'sample2@sample.com'),
44 (3, '情研二郎', 17, 'sample3@sample.com'),
45 (4, '日大花子', 49, 'sample4@sample.com');
46
```

3-2: エクスポートとインポート

そう、これはDB全体をSQL文として保存しておくことで
ここから復元することが可能なのです！！

これで万が一 データが消し飛んだとしても安心！！

3-2: エクスポートとインポート

インポートもすごく簡単！！

「エクスポート」タブの右隣にある「インポート」タブをクリックして
[ファイルを選択]ボタンをクリック！

バックアップしたsqlファイルを選択してあげるだけで復元完了！

3-3: PHPからデータベースを操作しよう！

Hello! DB!

3-3: PHPからデータベースを操作しよう！

はい、ここまでクソ長かったですね。お疲れ様です。
これからやっとプログラミングです。

いままではデータベースに慣れてもらうためにGUIツールを使っていました。

3-3: PHPからデータベースを操作しよう！

いい感じにデータベースとSQLに慣れてきたと思いますので、
PHPに入っていきます。

エディタを立ち上げて、新しくディレクトリを作ってください！

3-3: PHPからデータベースを操作しよう！

まずはPHPでデータベースにアクセスしてみましょう！
以下のコードを入力してみましょう！

3-3: PHPからデータベースを操作しよう！

データベースに接続ができると「接続に成功しています」と表示するPHPです

データベースの接続は ほぼ定型文となっています。

```
01_connect.php ×  
learn > 06_database > 01_connect.php > ...  
1  <?php  
2  
3  $dsn = 'mysql:dbname=sample;host=localhost;charset=utf8';      // DBの情報を設定  
4  // mysqlを使用し、dbname(DB名)はsample、host(ドメイン名)はlocalhost  
5  
6  $user = 'root';                                              // ユーザー名(本番ではrootは使わない)  
7  $password = '';                                            // パスワードは未設定のため空  
8  
9  
10 try{  
11     $dbh = new PDO($dsn, $user, $password);  
12     $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
13     echo '接続に成功しています';  
14  
15 }catch (PDOException $e){  
16     print('Connection failed: ' . $e->getMessage());  
17     die();  
18 }
```

3-3: PHPからデータベースを操作しよう！

データベースに接続ができると「接続に成功しています」と表示するPHPです

データベースの接続は ほぼ定型文となっています。

```
01_connect.php ×  
learn > 06_database > 01_connect.php > ...  
1  <?php  
2  
3  $dsn = 'mysql:dbname=sample;host=localhost;charset=utf8';      // DBの情報を設定  
4  // mysqlを使用し、dbname(DB名)はsample、host(ドメイン名)はlocalhost  
5  
6  $user = 'root';                                              // ユーザー名(本番ではrootは使わない)  
7  $password = '';                                            // パスワードは未設定のため空  
8  
9  
10 try{  
11     $dbh = new PDO($dsn, $user, $password);  
12     $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
13     echo '接続に成功しています';  
14  
15 }catch (PDOException $e){  
16     print('Connection failed:' . $e->getMessage());  
17     die();  
18 }
```

3-3: PHPからデータベースを操作しよう！

接続に成功すると、このように表示されます。



3-3-1 MEMO: ROOTユーザーって何だ！？

総合的な管理者かつ最も権限を持ったユーザーのことを

「ルートユーザー」 といいます。

Android OSで定番となっている「root化」もここから来ています。

あまりに権限が強すぎて、データの削除や変更などが簡単にできてしまうため、本番環境では使用されません。あくまで簡易的な仮のユーザーだと思ってください。

ちなみに合宿本番でもパスワードなどが各班に配布されます(予定)

\$passwordにはそれを入れてください。\$hostにはURLの部分を入れてね！

3-3-2: SQLをPHPで使ってみよう！

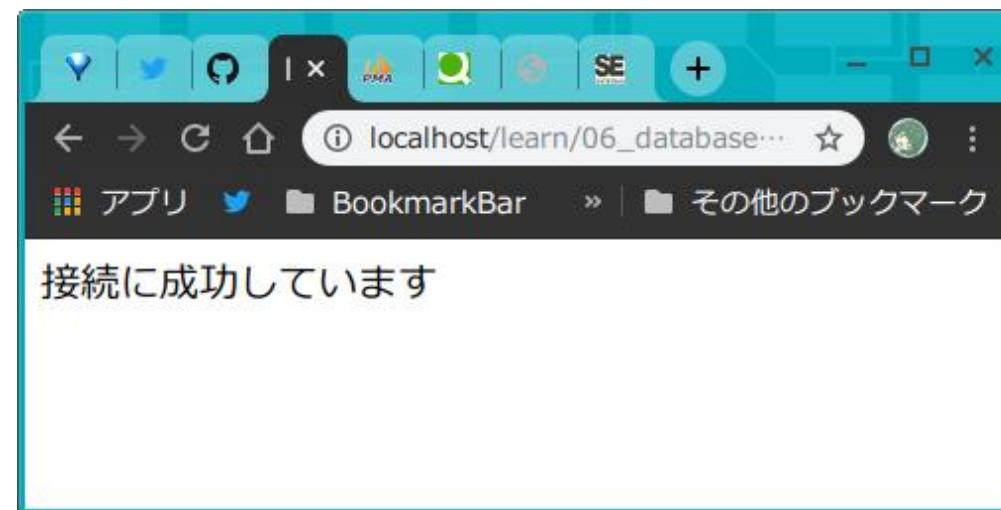
SQLもPHPから使用可能です。以下のコードを入力してみてください。
基本的には接続のコードと同じなのでコピペしてから編集のほうが楽です。

```
02_insert.php ×  
learn > 06_database > 02_insert.php > ...  
10 try{  
11     $dbh = new PDO($dsn, $user, $password);  
12     $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
13     // SQL文を格納する  
14     $sql = "INSERT INTO user (id, name, age, email) VALUES (NULL, '情研之助', '56', 'sample5@sample.com')";  
15     // SQLを実行する  
16     $stmt = $dbh->prepare($sql);  
17     $stmt->execute();  
18  
19     echo '接続に成功しています';  
20 }
```

3-3-2: SQLをPHPで使ってみよう！

今のコードはデータの挿入でした。

では、一度ブラウザで実行してから正常に追加が行われたか見てみましょう。



3-3-2: SQLをPHPで使ってみよう！

正常に追加されています！成功です！

+ オプション

	←→	▼	id	name	age	email
<input type="checkbox"/>			2	情研太郎	18	sample2@sample.com
<input type="checkbox"/>			3	情研二郎	17	sample3@sample.com
<input type="checkbox"/>			4	日大花子	49	sample4@sample.com
<input type="checkbox"/>			5	情研之助	56	sample5@sample.com

↑ Check all チェックしたものを:

3-3-3: 変数をSQLに当てはめよう！

```
❶ 03_insertVariable.php ×
learn > 06_database > ❷ 03_insertVariable.php > ...
1  <?php
2
3  $dsn = 'mysql:dbname=sample;host=localhost;charset=utf8';      // DBの情報を設定
4  // mysqlを使用し、dbname(DB名)はsample、host(ドメイン名)はlocalhost
5
6  $user = 'root';                                              // ユーザー名(本番ではrootは使わない)
7  $password = '';                                             // パスワードは未設定のため空
8
9  // 挿入するデータ
10 $name = '情研三郎';
11 $age = 6;
12
13 try{
14     $dbh = new PDO($dsn, $user, $password);
15     $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
16     // SQL文を格納する(置き換える部分をコロン+名前で指定)
17     $sql = "INSERT INTO user (name, age) VALUES (:name, :age)";
18
19     $stmt = $dbh->prepare($sql);
20
21     // SQLに変数を当てはめる
22     $stmt->bindValue(':name', $name, PDO::PARAM_STR);
23     $stmt->bindValue(':age', $age, PDO::PARAM_INT);
24
25     // 実行
26     $stmt->execute();
27     echo '処理が完了しました';
28
29 }catch(PDOException $e){
```

変数もSQLに当てはめることができます！

文字列の場合、PARAM_STR (22行目)
数字の場合、PARAM_INT(23行目)を
指定しよう！

SQL文で変数に置き換える部分を
:(コロン)+名前の形式にするのも
大事なポイント！

3-3-3: 変数をSQLに当てはめよう！

少し複雑ですが一緒に読み解いていきましょう。

\$dbhにはデータベースに接続するための切符(正確にはオブジェクト)が入っています。

\$dbh->setAttributeという書き方でエラーレポートの方法を指定できます。
PDO::ERRMODE_EXCEPTIONでデータベース操作中の問題の内容を受け取れます。

他にはPDO::ERRMODE_SILENTというオプションがあり、こちらはエラーの報告をしません。本番サイトで使用します。

3-3-3: 変数をSQLに当てはめよう！

SQLの書き方にも特徴があります。

VALUE (:name, :age)とすることで後で変数の値を置き換えることができるようになります。

SQLはすぐに実行せず、\$dbh->prepare(\$sql)とすることで変数の当てはめを待機する状態にします。これを「**プリペアドステートメント**」といいます。

\$stmt->bindValueで変数の値がSQL文に当てはめられます。

3-3-3: 変数をSQLに当てはめよう！

最後にエラー処理について。

try{ }catchと書くことによってデータベースエラーの対処ができます。
try以降で発生したエラーは例外処理をします。

```
29 }catch (PDOException $e){ // 例外を検知する
30     print('Connection failed:'. $e->getMessage()); // 例外を表示する
31     die(); // 処理を停止する
32 }
```

3-3-4MEMO: プリペアドステートメント

入力値をSQL文に当てはめる場合は必ずプリペアドステートメントを使用してください。一応直接SQL文に変数を当てはめることも可能なのですが、ここではその方法は紹介しません。普通に危ないので……

えっと、入力値をそのままSQLに代入してしまうと、悪意のあるコードが入力された場合、「**SQLインジェクション**」攻撃の危険性があります。

プリペアドステートメントをすることで悪意のある入力値を無害なものに変換してくれるのでです。

3-4: 取得したデータを表示しよう！

Hello! DB!

3-4: 取得したデータを表示しよう！

最初のconnect.phpをコピーして編集し、データを取得するコードを作ろう

```
07_select.php ×
learn > 06_database > 07_select.php > ...
1  <?php
2
3  $dsn = 'mysql:dbname=sample;host=localhost;charset=utf8';      // DBの情報を設定
4  // mysqlを使用し、dbname(DB名)はsample、host(ドメイン名)はlocalhost
5
6  $user = 'root';                                              // ユーザー名(本番ではrootは使わない)
7  $password = '';                                             // パスワードは未設定のため空
8
9
10 try{
11     $dbh = new PDO($dsn, $user, $password);
12     $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
13     $sql = "SELECT * FROM user";           // 全件取得する
14     $stmt = $dbh->prepare($sql);
15     $stmt->execute();
16     $data = array();                      // 配列を宣言
17     $count = $stmt->rowCount();          // 件数を取得する
18     while($row = $stmt->fetch(PDO::FETCH_ASSOC)){ // 一行ずつデータを取得
19         $data[] = $row;                  // 配列$dataに行を格納していく
20     }
21     echo '接続に成功しています<BR>';
22
23 }catch (PDOException $e){
24     print('Connection failed:' . $e->getMessage());
25     die();
26 }
27 var_dump($count);
28 var_dump($data);    // $dataの中身を確認
```

3-4: 取得したデータを表示しよう！

ブラウザで確認してみよう！すごく見づらいよね……

接続に成功しています

```
int(4) array(4) { [0]=> array(4) { ["id"]=> string(1) "2" ["name"]=> string(12) "情研太郎"
["age"]=> string(2) "18" ["email"]=> string(18) "sample2@sample.com" } [1]=> array(4) {
["id"]=> string(1) "3" ["name"]=> string(12) "情研二郎" ["age"]=> string(2) "17" ["email"]=>
string(18) "sample3@sample.com" } [2]=> array(4) { ["id"]=> string(1) "4" ["name"]=> string(12)
"日大花子" ["age"]=> string(2) "49" ["email"]=> string(18) "sample4@sample.com" } [3]=>
array(4) { ["id"]=> string(1) "5" ["name"]=> string(12) "情研之助" ["age"]=> string(2) "56"
["email"]=> string(18) "sample5@sample.com" } }
```

3-4: 取得したデータを表示しよう！

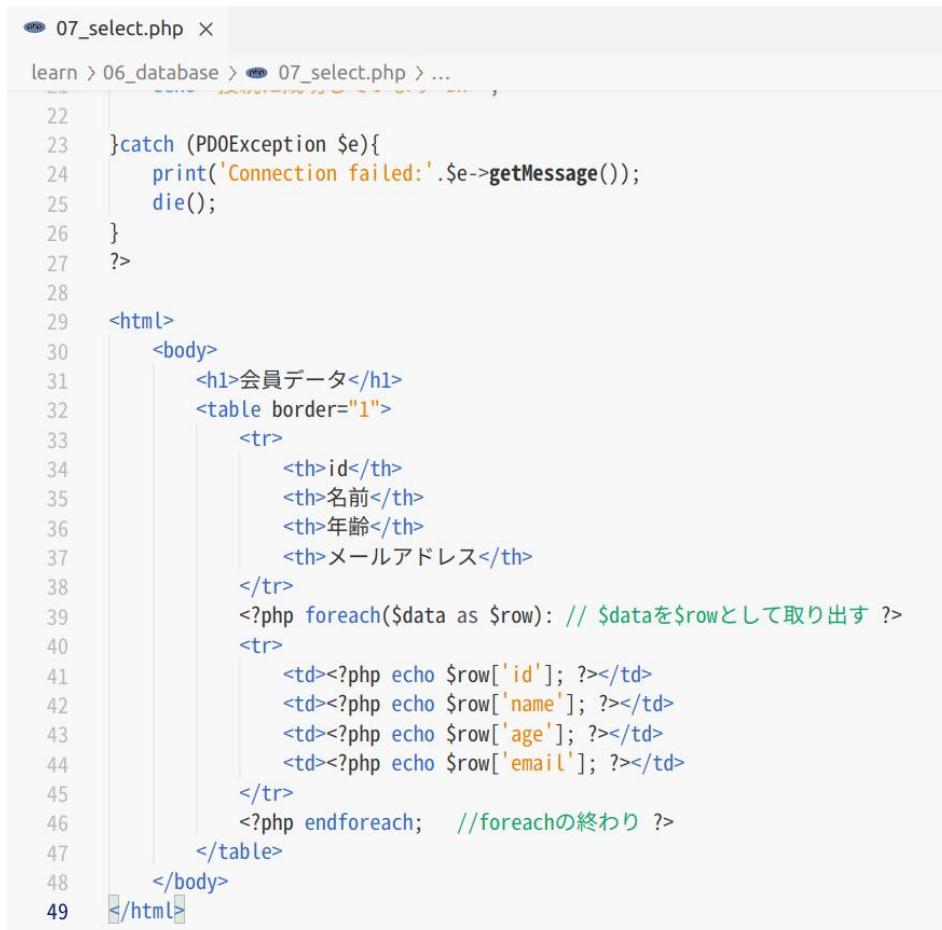
でも大丈夫！F12で「デベロッパーツール」を立ち上げよう！

実は改行コードが含まれるのでデベロッパーツールでは改行されて表示されるのです！

```
.▼<body> == $0
"接続に成功しています"
<br>
"int(4)
array(4) {
    [0]=>
    array(4) {
        ["id"]=>
        string(1) "2"
        ["name"]=>
        string(12) "情研太郎"
        ["age"]=>
        string(2) "18"
        ["email"]=>
        string(18) "sample2@sample.com"
    }
    [1]=>
    array(4) {
        ["id"]=>
        string(1) "3"
        ["name"]=>
        string(12) "情研二郎"
        ["age"]=>
        string(2) "17"
        ["email"]=>
        string(18) "sample3@sample.com"
    }
}
```

3-4: 取得したデータを表示しよう！

では、データが連想配列の形式になっていることが確認できたので、今のコードを編集して以下のように追記しよう！`var_dump()`は消してください。



```
07_select.php ×
learn > 06_database > 07_select.php > ...
22 }catch (PDOException $e){
23     print('Connection failed:' . $e->getMessage());
24     die();
25 }
26 ?>
27
28
29 <html>
30     <body>
31         <h1>会員データ</h1>
32         <table border="1">
33             <tr>
34                 <th>id</th>
35                 <th>名前</th>
36                 <th>年齢</th>
37                 <th>メールアドレス</th>
38             </tr>
39             <?php foreach($data as $row): // $dataを$rowとして取り出す ?>
40             <tr>
41                 <td><?php echo $row['id']; ?></td>
42                 <td><?php echo $row['name']; ?></td>
43                 <td><?php echo $row['age']; ?></td>
44                 <td><?php echo $row['email']; ?></td>
45             </tr>
46             <?php endforeach; //foreachの終わり ?>
47         </table>
48     </body>
49 </html>
```

3-4: 取得したデータを表示しよう！

書けたらブラウザで確認しましょう。

下の画像のように表示されていれば成功です！

接続に成功しています

会員データ

id	名前	年齢	メールアドレス
2	情研太郎	18	sample2@sample.com
3	情研二郎	17	sample3@sample.com
4	日大花子	49	sample4@sample.com
5	情研之助	56	sample5@sample.com

3-4-1 MEMO: {}の別な記述方法！？

foreachは通常、以下のように書きます。今までやってきたことですね。

```
foreach($data as $row){  
    // 繰り返し処理  
}
```

しかし、以下の場合、少し読みにくいコードになってしまいます。

```
foreach($data as $row){  
    if(条件式){  
        // 条件に応じた処理  
    }  
}
```

そのため、foreachもifも別の記述方法があるのです！！

3-4-1 MEMO: {}の別な記述方法！？

foreachは通常、以下のように書きます。今までやってきたことですね。

```
foreach($data as $row){  
    // 繰り返し処理  
}
```

しかし、以下の場合、少し読みにくいコードになってしまいます。

```
foreach($data as $row){  
    if(条件式){  
        // 条件に応じた処理  
    }  
}
```

そのため、foreachもifも別の記述方法があるのです！！

3-4-1 MEMO: {}の別な記述方法！？

このように書くことで、「}」が何の終わりなのか悩む必要がなくなります！
やったあ！

```
foreach($data as $row):
    if(条件式):
        // 条件に応じた処理
    endif;
endforeach;
```

4: おわりに

さて、データベース編もこれで終わり。

本当はPHP編があと一回あったはずなんですが
私の時間の使い方がど下手クソでつくれませんでした。
ごめんなさい。

さて、次は合宿です！
何かわからないことがあったら運営の人たちが教えてくれるので
じゃんじゃん質問してください！