# SQLite3の使い方（PDOクラス編）

## PDOクラスとは

データベースには、MySQL、PostgresQL、SQLite3などがあるが、それぞれ独自の関数をもっていて、ユーザーは使い分ける必要がある。しかし、PDOクラスを使えば、PDOがデータベースの差異を吸収してくれるので、ユーザーはどのデータベースを使っていても、同じ記述でよい。

## データベース作成と接続

$db = new PDO(‘sqlite:<データベール名>’);

例）test.dbというデータベースを作成するなら、

$db = new PDO(‘sqlite:test.db’);

test.dbがなかったら、新規作成して、接続してくれる。もし、すでにtest.dbがあったら、それに接続してくれる。

$dbというのは、単なる変数じゃなくて、データベースの様々な機能をもった「オブジェクト」というもの。ロボットのようなもの。

## データベースの切断

$db = null;

ただ、デフォルトではスクリプトの終了時に自動的に接続は切断されるので、省略できる。（設定によって、スクリプトが終了しても切断しないようにすることはできる）。

## テーブルの作成

**CREATE TABLE テーブル名(カラム名, ...) ;**

テーブル（表）を作成します。以下のような表を考えます。

|  |  |  |  |
| --- | --- | --- | --- |
| ID | name | age | address |
| 1 | Tanaka | 34 | Nara |
| 2 | Suzuki | 25 | Osaka |
| 3 | Kimura | 41 | Kyoto |

以下のクエリ文を作成します。(各カラムは 「,」(カンマ)で区切る）

$query = “create table person ( id, name, age, address )” ;

各カラムにデータ型を指定することもできます。

$query = “create table person ( id integer, name text, age integer, address test )”;

また、すでにテーブルが存在することも考慮して、「もしテーブルが存在しないならば」という条件を入れることもできます。

$query = “create table if not exists person ( id, name, age, address );

そのあとに、クエリ文をセットします。

$stmt = $db->query($query);

クエリ文を実行します。

$stmt->execure();

データベースへの接続からまとめて記述すると、以下のようになります。

<?php

$db = new PDO(‘sqlite:test.db’);

$query = “create table if not exists person ( id, name, age, address )”;

$stmt = $db->query($query);

$stmt->execute();

$db = null;

## データの入力

**INSERT INTO テーブル名 VALUES ( データ, ... ) ;**

データを入力するクエリ文を作成します。

$query = insert into person values ( 1, ‘Tanaka’, 34, ‘Nara’ ) ;

$stmt = $db->query($query);

$stmt->execute();

ただし、たいていの場合、値をそのまま入れることは少なくて、値をセットされた変数のかたちで入れます。たとえば、以下のように変数がセットされていたとします。

$id = 1;

$name = ‘Tanaka’;

$age = 34;

$address = ‘Nara’;

この場合は、次のようになります。

$query = insert into person values ( $id, $name, $age, $address );

$stmt = $db->query($query);

$stmt->execute();

しかし、これはセキュリティ面で危険なやり方です。以下のようにやることが推奨されています。

$query = insert into person values ( ?, ?, ?, ? );

$stmt = $db->prepare($query);

$stmt->bindValue(1, $id, PDO::PARAM\_INT);

$stmt->bindValue(2, $name, PDO::PARAM\_STR);

$stmt->bindValue(3, $age, PDO::PARAM\_INT);

$stmt->bindValue(4, $address, PDO::PARAM\_STR);

$stmt->execute();

あるいは、次のように、? ではなく、ラベルを使うやり方です。

$query = insert into person values ( :id, :name, :age, :address );

$stmt = $db->prepare($query);

$stmt->bindValue(‘:id’, $id, PDO::PARAM\_INT);

$stmt->bindValue(‘:name’, $name, PDO::PARAM\_STR);

$stmt->bindValue(‘:age’, $age, PDO::PARAM\_INT);

$stmt->bindValue(‘:address’, $address, PDO::PARAM\_STR);

$stmt->execute();

まとめて記述すると、以下のようになります。

<?php

function insertData ( $id, $name, $age, $address ) {

$db = new PDO(‘sqlite:test.db’);

$query = “insert into person values ( ?, ?, ?, ? )”;

$stmt = $db->prepare($query);

$stmt->bindValue(1, $id, PDO::PARAM\_INT);

$stmt->bindValue(2, $name, PDO::PARAM\_STR);

$stmt->bindValue(3, $age, PDO::PARAM\_INT);

$stmt->bincValue(4, $address, PDO::PARAM\_STR);

$stmt->execute();

$db = null;

return true;

}

$id = 1;

$name = ‘Tanaka’;

$age 34;

$address = ‘Nara’;

insertData( $id, $name, $age, $address );

$id = 2;

$name = ‘Suzuki’;

$age 25;

$address = ‘Osaka’;

insertData( $id, $name, $age, $address );

$id = 3;

$name = ‘Kimura’;

$age 41;

$address = ‘Kyoto’;

insertData( $id, $name, $age, $address );

## データの表示

**SELECT \* FROM テーブル名;**

入力したデータを表示してみます。次のクエリ文を作成してください。

$query = “select \* from person”;

以下のようなコードになります。

<?php

$db = new PDO(‘sqlite:test.db’);

$query = “select \* from person”;

$stmt = $db->query($query);

$stmt->execute();

$row = $stmt->fetch(PDO::FETCH\_ASSOC); --- (1)

var\_dump($row);

$db = null;

以下のように表示されます。

Array(4) {

[“id”]=>string(1) “1”

[“name”]=>string(6) “Tanaka”

[“age”]=> string(2) “34”

[“address”]=>string(4) “Nara”

}

連想配列になっていることがわかります。

そこで、データがある限り表示させるということで、上の (1) 以下を次のようにします。

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC);

$id = $row[‘id’];

$name = $row[‘name’];

$age = $row[‘age’];

$address = $row[‘address’];

echo ‘id:’, $id, ‘ 名前:’, $name, ‘ 年齢:’, $age, ‘ 住所:’, $address, “<br>\n”;

}

$db = null;

以下のように出力されます。

id:1 名前:Tanaka 年齢:34 住所:Nara<br>

id:2 名前:Suzuki 年齢:25 住所:Osaka<br>

id:3 名前:Kimura 年齢:41 住所:Kyoto<br>

## カラムを指定して入力

**INSERT INTO テーブル名(カラム名) VALUES(値);**

あるいは

**INSERT INTO テーブル名(カラム名1, カラム名2, ...) VALUES(値1, 値2, ...);**

次のデータを入力します。

|  |  |  |  |
| --- | --- | --- | --- |
| ID | name | age | address |
| 4 | Saito | 38 |  |

address欄はまだ未定です。次のクエリ文を作成します。

$query = “insert into person (id, name, age) values (:id, :name, :age )”;

以下のようなコードになります。

<?php

$id = 4;

$name = ‘Saito’;

$age = 38;

$db = new PDO(‘sqlite:test.db’);

$query = “insert into person ( id, name, age ) values ( :id, :name, :age )”;

$stmt = $db->prepare($query);

$stmt->bindValue(‘:id’, $id, PDO::PARAM\_INT);

$stmt->bindValue(‘:name’, $name, PDO::PARAM\_STR);

$stmt->bindValue(‘:age’, $age, PDO::PARAM\_INT);

$stmt->execute();

$db = null;

このコードを走らせてみて、そのあと、> sqlite3 test.db <Enter> でみてみます。

> sqlite3 test.db <Enter>

sqlite> select \* from person; <Enter>

1|Tanaka|34|Nara

2|Suzuki|25|Osaka

3|Kimura|41|Kyoto

4|Saito|38|

データが入っているのが、確認できます。

## データの更新

**UPDATE テーブル名 SET カラム名 = データ WHERE 条件式 ;**

その後、Saitoさんのaddressが判明したので、修正します。

<?php

$address = ‘Hyogo’;

$name = ‘Saito’;

$db = new PDO(‘sqlite:test.db’);

$query = “update person set address = :address where name = :name”;

$stmt = $db->prepare($query);

$stmt->bindValue(‘:address’, $address, PDO::PARAM\_STR);

$stmt->bindValue(‘:name’, $name, PDO::PARAM\_STR);

$stmt->execute();

$db = null;

確認しましょう。

>sqlite3 test.db <Enter>

sqlite> select \* from person; <Enter>

id name age address

---------- ---------- ---------- ----------

1 Tanaka 34 Nara

2 Suzuki 25 Osaka

3 Kimura 41 Kyoto

4 Saito 38 Hyogo

## データの並べ替え

**SELECT \* FROM テーブル名 ORDER BY カラム名 [ASC | DESC] ;**

年齢の昇順に並べかえて表示します。

<?php

$db = new PDO(‘sqlite:test.db’);

$query = “select \* from person order by age asc”;

$stmt = $db->query($query);

$stmt->execute();

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

echo $row[‘id’], ‘ ‘, $row[‘name’], ‘ ‘, $row[‘age’], ‘ ‘, $row[‘address’], “<br>\n”;

}

$db = null;

このように表示されます。

2 Suzuki 25 Osaka

1 Tanaka 34 Nara

4 Saito 38 Hyogo

3 Kimura 41 Kyoto

## データの抽出

特定の文字列を含むデータを取得する

**SELECT \* FROM テーブル名 WHERE カラム名 LIKE パターン;**

addressがnaraの人を抽出します。

sqlite> select \* from person where address like 'nara';

id name age address

---------- ---------- ---------- ----------

1 Tanaka 34 Nara

条件に合うデータをそのカラムだけ表示する

**SELECT カラム名 FROM テーブル名 WHERE 条件式;**

ageが30以上の人をidとnameとage で表示します。

sqlite> select id, name, age from person where age >= 30 ;

id name age

---------- ---------- ----------

1 Tanaka 34

3 Kimura 41

4 Saito 38

## カラムのデータ型を指定してテーブルを作成する

SQLite3は、各カラムのデータ型を指定しなくても、入力されたデータをみて、SQLite3が適切なデータ型に変換してくれます。

しかし、データ型を指定したい場合もあります。

今度は、テーブルの作成時に、各カラムのデータ型を指定してみます。

次のようなデータを考えてみます。

|  |  |  |  |
| --- | --- | --- | --- |
| ID | name | age | height |
| 1 | Taro | 13 | 143.5 |
| 2 | Hanako | 15 | 153.0 |
| 3 | Jiro | 17 | 170.3 |

テーブルを作成するコマンドはこのようになります。

**CREATE TABLE テーブル名 ( カラム名 データ型 , ...... ) ;**

カラムのデータ型には以下のようなものが指定できます。

|  |  |
| --- | --- |
| TEXT | 文字列 |
| NUMERIC | その他（日付・真偽値など） |
| INTEGER | 整数 |
| REAL | 浮動小数点型 |
| NONE | 指定なし（入力データそのまま） |

また、id については、次のような指定を加えてみます。

**PRIMARY KEY** …… 重複した値は指定できない。データ型が INTEGER の場合は連番が設定される。

**AUTOINCREMENT** …… 一度使われた値は二度使われない

そこで、以下のようにテーブルを作成してみます。各行はEnterキーで改行してください。

sqlite> create table student (

...> id integer primary key autoincrement ,

...> name text ,

...> age integer ,

...> height real );

データを入力します。

sqlite> insert into student (name, age, height) values('Taro', 13, 143.5);

sqlite> insert into student (name, age, height) values('Hanako', 15, 153.0);

sqlite> insert into student (name, age, height) values('Jiro', 17, 170.3);

データを見てみます。

sqlite> select \* from student;

id name age height

---------- ---------- ---------- ----------

1 Taro 13 143.5

2 Hanako 15 153.0

3 Jiro 17 170.3

自動で id が入力されています。また、小数点もはいっています。

**SQLite3 主なコマンド**

|  |  |
| --- | --- |
| .databases | データベース一覧 |
| .tables | テーブル一覧 |
| .quit | SQLite3を終了 |
| .exit | SQLite3を終了 |
| .help | ヘルプ |
| .show | 設定一覧 |
| .mode list | 区切り文字でカラムを区切り1行で出力（デフォルト） |
| 〃 line | 各カラムごとに行を分けて出力 |
| 〃 column | カラムごとに左揃えで出力 |
| .header on | カラム名（見出し）を出力 |
| 〃 off | カラム名（見出し）を出力しない |

参考URL

SQLiteコマンドの使い方 <https://www.dbonline.jp/sqlite/sqlite_command/>

SQLite入門 <https://www.dbonline.jp/sqlite/>

# 付録：MySQLでデータベースを作成する

## ユーザーを作成する

XAMPPコントロールパネルのshellを開きます。

MySQLにRoot でログインします。

# mysql -u root -p

Enter password:

パスワードは設定していないはずですから、そのままEnterキーをおします。

次のコマンドを入力して、どんなデータベースがあるのか確認しておきます。

mysql> show databases;

ユーザーを追加します。このユーザーに管理させるデータベース名を決めておきます。

今回は、testdb というデータベース名にしておきます。このデータベース名がないことを確認しておいてください。

ユーザー名をtestuser。パスワードも testuser にしておきます。

（ユーザー名とパスワードを同じにしておいたら、パスワードを忘れることはありません。セキュリティ上は好ましくないですが、開発用のデータベースであれば、大事なデータは登録しないので、大丈夫です）

|  |  |
| --- | --- |
| データベース名 | testdb |
| ユーザー名 | testuser |
| パスワード | testuser |

以下のコマンドを入力します。

mysql> grant all on testdb.\* to 'testuser'@'localhost' identified by 'testuser';

Query OKと表示されたら、うまくいっています。

ログアウトします。

mysql> quit

Bye

## データベースを作成する

先ほど作成したユーザーでログインします。パスワードはユーザー名と同じです。

# mysql -u testuser -p

Enter password:

どんなデータベースがあるか見てみます。

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

+--------------------+

information\_schema は、もとからあるデータベースです。つまり、まだ何も作っていません。

では、データベースを作成します。

mysql> create database testdb;

Query OK, 1 row affected (0.00 sec)

これでデータベースが作成できています。

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| testdb |

+--------------------+