第８回　データベース接続（SELECT）

８－１．データベース接続

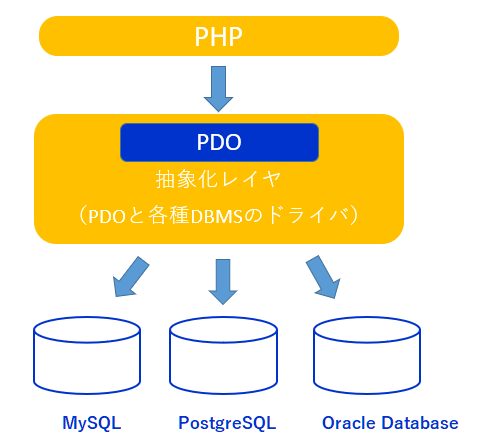
８－１－１．PDOとは

PHPからSQLを実行するために、PDO（PHP Data Object）を使用します。

PDOとは、PHPの拡張モジュールです。データベース製品の違いを吸収するための統一されたインターフェース（PDOクラス）を提供します。

　従来は様々なデータベース（postgreSQL, Microsoft SQL Server, Oracle Database）などにより、プログラムの書き方が異なりました。そのため、利用するデータベースを変更すると、PHPプログラムも大幅に修正する必要がありました。

　このような手間を省くため、PHPとデータベース（正確にはDBMS）の間に抽象化レイヤを挟んで、各種DBMSの違いを抽象化レイヤで吸収し、DBMSが異なっても同じ処理をできるようにするというのが、PDOの役割です。抽象化レイヤには、PDOとPDOが使用する各種ドライバが含まれています。



８－１－２．データベースへの接続

データベース処理は下記のような手順になります。Sampleコードと併せて確認していきましょう。

1. PDOクラスをインスタンス化する
2. PDOの動作オプションを指定する
3. SQL文の準備と実行
4. SQL実行結果の処理
5. PDOオブジェクトを破棄

★Sampleコード（SELECT　＊条件指定なしの場合）

Sampleのデータベース情報**（皆さんが使用するDBとは違うものを使用しています）**

・ホスト名　　　　　：　localhost

・DB名　　　　　　：　malldb

・DBユーザー　　：　malluser

・DBパスワード　：　mall

・テーブル 　：products



1. ①
2. PDOインスタンス化

■構文

C

B

A

new PDO([接続先DB情報], [DBのログインID], [DBのパスワード])

コンストラクタの引数は、３つです。

A

　接続先DB情報を表す文字列を、DSN（Data Source Name）といいます。

データベースにより書式が異なりますが、MySQLの場合は下記のようになります。

mysql:host=[接続先ホスト名またはIPアドレス]; dbname=[接続先DB名]; charset=[接続時の文字エンコーディング名]

＊緑色部分には環境に合わせた値を記載します。

B

　DBのログインIDを記載します。

C

　DBのログインIDを記載します。

1. PDOの動作オプションを指定

■構文（PDO::setAttributeメソッド）

$インスタンス変数->setAttribute([オプション種別], [オプション値])

■setAttributeメソッドで指定できるオプション定数

|  |  |  |  |
| --- | --- | --- | --- |
| オプション種別 | 指定方法 | 意味 | デフォルト値 |
| PDO:ATTR\_CASE | 定数で指定 | SELECT結果のカラム名について大文字／小文字の扱いを指定 | PDO::CASE\_NATURAL |
| PDO::ATTR\_ERRMODE | 定数で指定 | エラー通知方法を指定 | PDO::ERRMODE\_SILENT |
| PDO::ATTR\_ORACLE\_NULLS | 定数で指定 | NULLと空文字の変換方法を指定。Oracle以外でも使用可能 | PDO::NULL\_NATURAL |
| PDO::ATTR\_AUTOCOMMIT | 真偽値で指定 | 自動コミットするか否かを指定 | true |
| PDO::ATTR\_EMULATE\_PREPARES | 真偽値で指定 | プリペアドステートメントのエミュレーションを有効にするか否かを指定 | true |
| PDO::ATTR\_DEFAULT\_FETCH\_MODE | 定数で指定 | SELECT結果のPHO変数へのマッピング方法を指定 | PDO::FETCH\_BOTH |

■PDO：：ATTR\_CASEに指定できる値

|  |  |
| --- | --- |
| 定数名 | 意味 |
| PDO:CASE\_LOWER | 小文字に変換する |
| PDO::CASE\_NATURAL | 変換しない |
| PDO::CASE\_UPPER | 大文字に変換する |

■PDO::ATTR\_ERRMODEに指定できる値

|  |  |
| --- | --- |
| 定数名 | 意味 |
| PDO::ERRMODE\_SILENT | エラーを出力しない。PDO::errCodeメソッドでエラーコード取得のみ可能 |
| PDO::ERRMODE\_WARNING | E\_WARNING定数レベルのエラーを出力 |
| PDO::ERRMODE\_EXCEPTION | 例外（おもにＰＤＯException）をスローする |

■PDO::ATTR\_ORACLE\_NULLSに指定できる値

|  |  |
| --- | --- |
| 定数名 | 意味 |
| PDO::NULL\_NATURAL | 変換しない |
| PDO::NULL\_EMPTY\_STRING | 空文字をNULLに変換する |
| PDO::NULL\_TO\_STRING | NULLを空文字に変換する |

■PDO::ATTR\_DEFAULT\_FETCH\_MODEに指定できる値

|  |  |  |
| --- | --- | --- |
| 定数名 | 意味 | 結果セットの変数の参照例 |
| PDO::FETCH\_ASSOC | カラム名をキーとした連想配列を返す | $result[‘column’] |
| PDO::FETCH\_NUM | 0で始まる配列を返す | $result[0] |
| PDO::FETCH\_BOTH | FETCH\_ASSOCとFETCH\_NUMをミックスした配列を返す | $result[0] / $result[‘column’] |
| PDO::FETCH\_OBJ | カラム名をプロパティに持つインスタンスを返す | $result->column |

1. SQL文の準備と実行

■構文（PDO::prepareメソッド） SQLの準備

$ PDOステートメント変数 = $インスタンス変数->prepare([SQL])

PHPの変数値を入れる箇所は、プレースホルダとして設定しておきます。そのまま文字列に変数名を連結しないようにしてください。

例：＊赤字の箇所がプレースホルダです。冒頭にコロンを付けることでプレースホルダを表します。

$ PDOステートメント変数 = $インスタンス変数->prepare(‘SELECT \* FROM PRODUCT WHERE NAME = :name’);

プレースホルダとは、仮に確保された場所のことで、後ほど何らかの値をセットすることになります。

■構文　プレースホルダに値をバインドする

$PDOステートメント変数->bindParam([プレースホルダ名], [バインドする値], [値のデータ型])

■値のデータ型に指定できる定数

|  |  |
| --- | --- |
| 定数名 | 意味 |
| PDO::PARAM\_BOOL | 真偽値型 |
| PDO::PARAM\_NULL | NULL |
| PDO::PARAM\_INT | 整数型 |
| PDO::PARAM\_STR | 文字列型、または小数型 |
| PDO::PARAM\_LOB | ラージオブジェクト型 |

■構文（PDOStatement::executeメソッド） SQLの実行

$ PDOステートメント変数->execute( )

1. SQL実行結果の処理

SELECTの場合、実行結果をfetchメソッドまたはfetchAllメソッドにて取得します。

SQLを実行すると、結果セット（メモリ上に作成された仮想的なテーブル）が作成されます。フェッチとは結果セットから1レコードっまたは全レコード分を取り出してPHPの変数に割り当てることです。

■構文（PDOStatement::fetchメソッド）

$ PDOステートメント変数->fetch([フェッチモード定数])

　　　　　　　　　または

$ PDOステートメント変数->fetchAll([フェッチモード定数])

\*フェッチモード定数に指定できる値は、「■PDO::ATTR\_DEFAULT\_FETCH\_MODEに指定できる値

」と同じです。

fetchメソッドの方がPHPのメモリ使用量を大幅に抑えられます。

Sampleコードでは、1件ずつ取得した結果を、変数$resultに代入しています。

1. PDOオブジェクトを破棄

DBの処理が終了したら、接続を閉じましょう。接続の順番は以下の通りです。

1. PDOステートメント変数を閉じるにはnullを代入します。
2. PDOインスタンス変数を閉じるにはnullを代入します。

＊明示的にこれを行わなかった場合は、スクリプトの終了時に自動的に 接続が閉じられます。