

# TODOリスト

2020/01/27

追加

期限日	項目	未完了	完了	削除	
<del>2020-01-01</del>	<del>元旦なので初詣に行く</del>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/>	実行
<del>2020-01-07</del>	<del>ワークで難波神社へ初詣に行く</del>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/>	実行
2020-02-03	ITワーク・PHP開発者のためのBootstrap4入門	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	実行

# TODOリスト・仕様(1)

- 個人で使う
- ログイン機能なし
- 入力項目
  - ✓ 期限日  
デフォルト値：今日の日付
  - ✓ TODO項目  
デフォルト値：空
  - ✓ 追加ボタン  
ボタンをクリックすると、TODO項目が追加される
- 表示項目
  - ✓ 期限日
  - ✓ TODO項目
  - ✓ 完了/未完了ラジオボタン
  - ✓ 削除チェックボックス

# TODOリスト・仕様(2)

- 実行ボタンを押すとき
  - ✓ 項目を完了するときは「完了」にチェックを入れる
  - ✓ 項目が完了の場合は、「期限日」「TODO項目」に取り消し線を入れる
  - ✓ 項目を未完了にするときは「未完了」にチェックを入れる
- 一覧表示をするとき
  - ✓ 項目が「完了」のときは、「完了」のラジオボタンが選択状態になっている
  - ✓ 項目が「未完了」のときは、「未完了」のラジオボタンが選択状態になっている。
  - ✓ 削除されている項目は表示しない。(is\_deleted=1のレコードは表示しない)
- 削除チェックボックス
  - ✓ 削除する場合はチェックを入れる
  - ✓ 削除したら一覧には表示されない
  - ✓ 実際に削除するのではなく、is\_deleted=1にする
- 実行ボタン
  - ✓ 「完了」および「削除」を実行する

# TODOリスト・画面遷移と処理内容

期限日	項目	未完了	完了	削除
2020-01-01	元旦なので初詣に行く	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/> 実行
2020-01-07	夕方で難波神社へ初詣に行く	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/> 実行
2020-02-03	ITワーク・PHP開発者のためのBootstrap4入門	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/> 実行

- index.php

1. テーブルにあるレコードを一覧表示

- add.php

1. index.phpからPOSTされたデータを受け取り
2. レコードをインサート
3. index.phpへリダイレクト

- action.php

1. index.phpから受け取ったデータを受け取り
2. 該当のレコードをアップデート
3. index.phpへリダイレクト

# TODOリスト・フォームと値の受け取り(1)

HTML

```
<form method="post" action="./add.php">  
  ~  
</form>
```

**method="post"**

フォームの値を送信する方法です。  
**post** と **get** があります。

**action="./add.php"**

フォームの値の送信先のURL（ファイル）です。

# TODOリスト・フォームと値の受け取り(2) フォームの部品

- テキストボックス

value属性に値を指定することで、初期値を設定できます。

```
<input type="text" name="item" value="やらなければならないこと">
```

- ラジオボタン

```
<input type="radio">
```

name属性を同じ名前にすることで、同じグループになります。

value属性で、送信する値を設定できます。

「checked」を記載することで、選択状態にすることができます。

```
<input type="radio" name="complete" value="0" checked>
```

```
<input type="radio" name="complete" value="1">
```

- チェックボックス

value属性で、送信する値を設定できます。

「checked」を記載することで、選択状態にすることができます。

```
<input type="checkbox" name="delete" value="1">
```

- 隠しフィールド

ブラウザの画面には表示されませんが、値を送ることができます。

```
<input type="hidden" name="id" value="1">
```

# TODOリスト・フォームと値の受け取り(2) 値の受け取り(1)

フォームから送信された値は、action先のPHPファイルで受け取ることができます。

## HTML

```
<input type="text" name="item" value="やらなければならないこと">
<input type="radio" name="complete" value="0" checked>
<input type="radio" name="complete" value="1">
<input type="checkbox" name="delete" value="1">
```

PHP (method="post"で送信したとき)

```
$name = $_POST['item'];
$complete = $_POST['complete'];
$delete = $_POST['delete'];
```

PHP (method="get"で送信したとき)

```
$name = $_GET['item'];
$complete = $_GET['complete'];
$delete = $_GET['delete'];
```

# TODOリスト・フォームと値の受け取り(3) 値の受け取り(2)

2020-02-29 うるう年なので2月は29日まであります ☒ ☐ ☐ 実行

チェックボックスにチェックが入っていないときは、何も送信されません。

HTML

```
<input type="checkbox" name="delete">
```

PHP

```
$delete = $_POST['delete']; // $_POST['delete']そのものが存在しないので、エラーが発生します。
```

isset()を使って、存在するかどうかを判定します。

<https://www.php.net/manual/ja/function.isset.php>

```
if (isset($_POST['delete'])) {  
    // true → $_POST['delete']が存在するときの処理  
} else {  
    // true → $_POST['delete']が存在しないときの処理  
}
```



# TODOリスト・データベースへの接続

```
// データベースに接続するための文字列（DSN 接続文字列）です。  
// MySQLに接続するときの書き方です。  
// dbname      データベース名  
// host        データベースサーバーのホスト名、またはIPアドレス  
//             XAMPPやMAMPを使うときは「localhost」  
// charset     データベースのデフォルトの文字コード。「utf8」にします。  
$dsn = 'mysql:dbname=todo_list;host=localhost;charset=utf8';  
  
// PDOクラスのインスタンスを作ります。  
// 引数は、上記のDSN、データベースのユーザー名、パスワードです。  
// XAMPPの場合はデフォルトでパスワードなし、MAMPの場合は「root」になっています。  
$dbh = new PDO($dsn, 'root', '');  
  
// エラーが起きたときのモードを指定します。  
// 「PDO::ERRMODE_EXCEPTION」を指定すると、エラー発生時に例外がスローされます。  
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

# TODOリスト・SQLの実行 SELECT文 (1)

```
// データベースに接続
$dsn = 'mysql:dbname=todo_list;host=localhost;charset=utf8';
$dbh = new PDO($dsn, 'root', '');
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// SQLのselect文です
$sql = 'select * from todo_items where is_deleted=:is_deleted';

// SQL文を実行する準備をします。
$stmt = $dbh->prepare($sql);

// SQL文の「:is_deleted」のところに、変数の値を割り当て（バインド）します。
// 変数の値が整数のときは、第3引数に「PDO::PARAM_INT」を設定します。
$stmt->bindValue(':is_deleted', $is_deleted, PDO::PARAM_INT);
```

# TODOリスト・SQLの実行 SELECT文 (2)

// SQL文を実行します。

```
$stmt->execute();
```

// select文は実行結果が連想配列で返却されますので、変数に代入します。

```
$list = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

// 連想配列の要素を1件ずつ取り出し、表示します。

```
foreach ($list as $v) {  
    echo $v['todo_item'];  
}
```

# TODOリスト・SQLの実行 insert、update、delete (1)

```
// データベースに接続
$dsn = 'mysql:dbname=todo_list;host=localhost;charset=utf8';
$dbh = new PDO($dsn, 'root', '');
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// SQLのインサート文です
$sql = '';
$sql .= 'insert into todo_items (';
$sql .= 'expiration_date,';
$sql .= 'todo_item';
$sql .= ') values (';
$sql .= ':expiration_date,';
$sql .= ':todo_item';
$sql .= ')';
```

# TODOリスト・SQLの実行 insert、update、delete (2)

```
// SQL文を実行する準備をします。  
$stmt = $dbh->prepare($sql);  
  
// SQL文の「:expiration_date」のところに変数の値を割り当て（バインド）します。  
// 文字列、日付などは、第3引数に「PDO::PARAM_STR」を設定します。  
$stmt->bindValue(':expiration_date', $expiration_date, PDO::PARAM_STR);  
  
// SQL文の「:todo_item」のところに変数の値を割り当て（バインド）します。  
// 文字列、日付などは、第3引数に「PDO::PARAM_STR」を設定します。  
$stmt->bindValue(':todo_item', $todo_item, PDO::PARAM_STR);  
  
// SQL文を実行します。  
// insert文の実行結果を受け取る必要はありません。  
// SQL文を実行したときにエラーが起きたら例外がスローされるためです。  
$stmt->execute();
```

※ insert文、update文、delete文、すべてSQL文の内容が変わるだけで、書き方は同じです。

# TODOリスト・SQLの実行 bindValueメソッド

// バインドする値の型がintegerのとき、第3引数に「 PDO::PARAM\_INT」を指定します。

```
$stmt->bindValue(':intValue', $intValue, PDO::PARAM_INT);
```

// バインドする値の型が文字列、日付型の文字列のとき、

// 第3引数に「 PDO::PARAM\_STR」を指定します。

```
$stmt->bindValue(':strValue', $strValue, PDO::PARAM_STR);
```

```
$stmt->bindValue(':dateValue', $dateValue, PDO::PARAM_STR);
```

# TODOリスト・例外の処理 (1)

例外が発生する可能性があるときは、`try { } catch() {}` で捕捉します。

```
try {  
    // データベース接続に失敗すると例外が発生（スロー）されます。  
    $dsn = 'mysql:dbname=todo_list;host=localhost;charset=utf8';  
    $dbh = new PDO($dsn, 'root', '');  
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
    $sql = 'insert into todo_items (~';  
    $stmt = $dbh->prepare($sql);  
    $stmt->bindValue(':intValue', $intValue, PDO::PARAM_INT);  
  
    // SQLの構文にミスがあるとSQL文の実行時に例外が発生（スロー）されます。  
    $stmt->execute();  
} catch (Exception $e) {  
    // 例外が発生（スロー）されると、処理がcatch()の中にジャンプします。  
    // $eの中にExceptionクラスのインスタンスが入ります。  
    // 例外発生時の処理をここに書きます。  
    var_dump($e);    // var_dump()で例外が発生した理由などが表示されます。  
    exit;  
}
```

# TODOリスト・例外の処理 (2)

参考)

例外をわざと発生させる事ができます。

```
try {  
    // 例外を発生（スロー）させます。  
    throw new Exception('エラーです');  
} catch (Exception $e) {  
    // 例外が発生（スロー）されると、処理がcatch()の中にジャンプします。  
    // $eの中にExceptionクラスのインスタンスが入ります。  
    // 例外発生時の処理をここに書きます。  
    var_dump($e);    // var_dump()で例外が発生した理由などが表示されます。  
    exit;  
}
```