PHP についての別紙「ソースコード」　　佐野　尚之

注意事項

・業務で PHP を使ったことがないため、コーディング方法 / セキュリティ対策 (XSS, SQL Injection など)に問題がある可能性があります。
あくまでサンプルです。

■sp_dbsetup.php・・・データベース、テーブル、初期データ作成
(c:¥xampp¥htdocs¥shopping)

```
<html lang="ja">
<head>
<head>
<meta http-equiv="content-language" content="ja">
<meta charset="shift_jis">
<title>sample.db の作成 及び 対象テーブルを作成する</title>
</head>
<body>
<p>sample.db の作成 及び 対象テーブルを作成...</p>
<?php
require_once('sp_dbmanager.php');

try {
    // データベース接続
    $db = getDb();

    //$sql = 'DROP TABLE product';
    //if (! $db->query($sql)) {
    //    echo 'Failed : ' . $sql;
    //}
    //$sql ='DROP TABLE member';
    //if (! $db->query($sql)) {
    //    echo 'Failed : ' . $sql;
    //}
    //$sql = 'DROP TABLE shopping_order';
```

```
    //if (! $db->query($sql)) {
    //    echo 'Failed : ' . $sql;
    //}

    $sql = 'CREATE TABLE product(code integer primary key, name text not null, price integer)';
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql ='CREATE TABLE member(code integer primary key, name text not null, address text not null, email text)';
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = 'CREATE TABLE shopping_order(code primary key, member integer, product integer, order_date date not null)';
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }

    $sql = "INSERT INTO product VALUES(1, 'みかん', 150)";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO product VALUES(2, 'りんご', 100)";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO product VALUES(3, 'ぶどう', 300)";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO product VALUES(4, 'ばなな', 150)";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
```

```
    $sql = "INSERT INTO member VALUES(1, '山田　太郎', '新宿', 'yamada@example.jp')";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO member VALUES(2, '山田　花子', '品川',  null)";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO member VALUES(4, '田中　次郎', '渋谷',  'tanaka@example.jp')";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }

    $sql = "INSERT INTO shopping_order VALUES(1, 1, 3, '2005-10-10')";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO shopping_order VALUES(2, 1, 1, '2005-10-14')";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
    $sql = "INSERT INTO shopping_order VALUES(3, 4, 2, '2005-11-15')";
    if (! $db->query($sql)) {
        echo 'Failed : ' . $sql;
    }
} catch (PDOException $e) {
    // エラーメッセージ
    $err = $db->errorInfo();
    die($err[2]);
}
// データベース切断
unset($db);
?>
<p>完了</p>
```

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

■sp_dbmanager.php・・・DB 接続(sp_dbsetup.php 用)
(c:¥xampp¥htdocs¥shopping)

```
<?php
function getDb() {
    try {
        // データベース sample に接続する
        $db = new PDO('sqlite:sample.db');
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        die('データベースに接続できません。' . $e->getMessage());
    }
    return $db;
}
```

■sp_smarty.php・・・Smarty 設定
(c:¥xampp¥htdocs¥shopping)

```
<?php
define('SMARTY_DIR','C:xampp/php/includes/smarty/');
require_once(SMARTY_DIR . 'Smarty.class.php');

$tpl = new Smarty;
$tpl->setTemplateDir('C:xampp/htdocs/shopping/templates');
$tpl->setCompileDir('C:xampp/htdocs/shopping/templates_c');
$tpl->setCacheDir('C:xampp/htdocs/shopping/cache');
$tpl->setConfigDir('C:xampp/htdocs/shopping/configs');
```

■sp_model.php・・・shopping クラス

(c:¥xampp¥htdocs¥shopping)

```
<?php
/**
* Shopping クラス
*/
class Shopping {
    private $db;
    private $product_code;
    private $member_code;
    private $next_code;

    /**
    * コンストラクタ
    */
    public function __construct() {
        // データベース接続
        $this->conectDb();
    }

    /**
    * デストラクタ
    */
    public function __destruct() {
        // データベース切断
        unset($db);
    }

    /**
    * プロダクトコードを取得
    */
    public function getProductCode() {
        return $this->product_code;
    }
```

```
/**
* プロダクトコードに値を設定
*/
public function setProductCode($product_code) {
    $this->product_code = $product_code;
}

/**
* メンバーコードを取得
*/
public function getMemberCode() {
    return $this->member_code;
}

/**
* メンバーコードに値を設定
*/
public function setMemberCode($member_code) {
    $this->member_code = $member_code;
}

/**
* ネクストコードを取得
*/
public function getNextCode() {
    return $this->next_code;
}

/**
* ネクストコードに値を設定
*/
public function setNextCode($next_code) {
    $this->next_code = $next_code;
}
```

```
/**
* データベース接続
*/
private function conectDb() {
    try {
        // データベース sample に接続する
        $this->db = new PDO('sqlite:sample.db');
        $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        die('データベースに接続できませんでした。' . $e->getMessage());
    }
}

/**
* プロダクトコードが product テーブルに存在するか確認
*/
public function productcodeCheck() {
    try {
        $sql = 'SELECT COUNT(*) AS CNT FROM product WHERE code = :product_code';
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':product_code', $this->product_code);
        $stmt->execute();
        while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            $cnt =  $row['CNT'];
        }
        if ($cnt < 1) {
           die('商品の選択が適切ではありません。');
        }
    } catch (PDOException $e) {
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* メンバーコードが member テーブルに存在するか確認
*/
public function membercodeCheck() {
    try {
        $sql = 'SELECT COUNT(*) AS CNT FROM member WHERE code = :member_code';
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':member_code', $this->member_code);
        $stmt->execute();
        while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            $cnt =  $row['CNT'];
        }
        if ($cnt < 1) {
            die('購入者の選択が適切ではありません。');
        }
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* product テーブルの全データを取得
*/
public function productAllData() {
    try {
        $sql = 'SELECT code AS product_code, name AS product_name,
                price AS product_price FROM product ORDER BY code';
        $stmt = $this->db->query($sql);
        $row = $stmt->fetchAll(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('データベースへの問い合わせに失敗しました。');
        }
        // 取得した値を返す
        return $row;
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* member テーブルの全データを取得
*/
public function memberAllData() {
    try {
        $sql = 'SELECT code AS member_code, name AS member_name
                FROM member ORDER BY code';
        $stmt = $this->db->query($sql);
        $row = $stmt->fetchAll(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('データベースへの問い合わせに失敗しました。');
        }
        // 取得した値を返す
        return $row;
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* プロダクトコードに対応する商品データを取得
*/
public function productData() {
    try {
        $sql = 'SELECT code AS product_code, name AS product_name,
                price AS product_price
                FROM product WHERE code = :product_code';
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':product_code', $this->product_code);
        $stmt->execute();
        $row = $stmt->fetchAll(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('[productData()]データベースへの問い合わせに失敗しました。');
        }
        // 取得した値を返す
        return $row;
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* メンバーコードに対応する商品データを取得
*/
public function memberData() {
    try {
        $sql = 'SELECT code AS member_code, name AS member_name
                FROM member WHERE code = :member_code';
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':member_code', $this->member_code);
        $stmt->execute();
        $row = $stmt->fetchAll(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('[memberData()]データベースへの問い合わせに失敗しました。');
        }
        // 取得した値を返す
        return $row;
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* shopping_order テーブルの code 例の最大値+1 を取得して設定
*/
public function shoppingorderData() {
    try {
        $sql = 'SELECT MAX(code) + 1 AS next_code FROM shopping_order';
        $stmt = $this->db->query($sql);
        $row = $stmt->fetch(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('[shoppingorderData()]データベースへの問い合わせに失敗しました。');
        }
        $this->next_code = $row['next_code'];
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
/**
* コードに対応するshoppingorderデータを取得
*/
public function shoppingorderData2($code) {
    try {
        $sql = 'SELECT code as shopping_code, member as shopping_member,
                product AS shopping_product, order_date AS shppping_order_date
                FROM shopping_order WHERE code = :shopping_code';
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':shopping_code', $code);
        $stmt->execute();
        $row = $stmt->fetchAll(PDO::FETCH_ASSOC); // 連想配列として取得
        if ( ! $row ) {
            die('[shoppingorderData2()]データベースへの問い合わせに失敗しました。');
        }
        // 取得した値を返す
        return $row;
    } catch (PDOException $e) {
        // エラーメッセージ
        $err = $this->db->errorInfo();
        die($err[2]);
    }
}
```

```
    /**
    * shopping_order テーブルにデータを登録
    */
    public function shoppingorderDataInsert() {
        try {
            $order_date = date('Y-m-d');

            // トランザクションの開始
            $this->db->beginTransaction();

            // shopping_order テーブルにデータを登録
            $sql = 'INSERT INTO shopping_order (code, member, product, order_date)
                    VALUES (:next_code, :member_code, :product_code, :order_date)';
            $stmt = $this->db->prepare($sql);
            $stmt->bindParam(':next_code', $this->next_code);
            $stmt->bindParam(':member_code', $this->member_code);
            $stmt->bindParam(':product_code', $this->product_code);
            $stmt->bindParam(':order_date', $order_date);
            $stmt->execute();

            // コミット
            $this->db->commit();
        } catch (PDOException $e) {
            // ロールバック
            $this->db->rollBack();
            // エラーメッセージ
            $err = $this->db->errorInfo();
            die($err[2]);
        }
    }
}
```

■sp_entry.php・・・商品と購入者選択画面

(c:¥xampp¥htdocs¥shopping)

```
<?php
require_once('sp_smarty.php');
require_once('sp_model.php');

// クラス生成
$shopping = new Shopping();

// $product_code に対応する商品データを取得
$arrValues = $shopping->productAllData();
$tpl->assign('productdata', $arrValues);

// $member_code に対応する商品データを取得
$arrValues = $shopping->memberAllData();
$tpl->assign('memberdata', $arrValues);

// 後処理
//$shopping = null;

// テンプレートを用いて表示
$tpl->display('sp_entry.tpl');
```

■sp_entry.tpl

(c:¥xampp¥htdocs¥shopping¥template)

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta http-equiv="content-language" content="ja">
<meta charset="shift_jis">
</head>
<body>
<form action="sp_confirm.php" method="post">
商品：<br>
<!--- {foreach from=$productdata item=var} --->
<input type="radio" name="product" value="{$var.product_code}">
{$var.product_name} ({$var.product_price}円)<br>
<!--- {/foreach} --->
<br>
購入者：
<select name="member">
<!--- {foreach from=$memberdata item=var} --->
<option value="{$var.member_code}">{$var.member_name}</option>
<!--- {/foreach} --->
</select>
<br><br>
<input type="submit" name="submit" value="購入">
</form>
</body>
</html>
```

■sp_confirm.php

(c:¥xampp¥htdocs¥shopping)

```
<?php
require_once('sp_smarty.php');
require_once('sp_model.php');

// entry.php より POST で受け取ったデータを変数に格納
$product_code = (isset($_POST['product'])) ? $_POST['product'] : '';
$member_code = (isset($_POST['member'])) ? $_POST['member'] : '';

// クラス生成
$shopping = new Shopping();

// $product_code が存在するか確認
$shopping->setProductCode($product_code);
$shopping->productcodeCheck();

// $member_code が存在するか確認
$shopping->setMemberCode($member_code);
$shopping->membercodeCheck();

// product テーブルのデータを取得
$shopping->setProductCode($product_code);
$arrValues = $shopping->productData();
for ($i = 0 ; $i < count($arrValues); $i++) {
    // $product_code に対応する商品データをテンプレートにセット
    $tpl->assign('product_code', $arrValues[$i]['product_code']);
    $tpl->assign('product_name', $arrValues[$i]['product_name']);
    $tpl->assign('product_price', $arrValues[$i]['product_price']);
}

// $member_code に対応する商品データを取得
$shopping->setMemberCode($member_code);
$arrValues = $shopping->memberData();
```

```
for ($i = 0 ; $i < count($arrValues); $i++) {
    // $member_code に対応する商品データをテンプレートにセット
    $tpl->assign('member_code', $arrValues[$i]['member_code']);
    $tpl->assign('member_name', $arrValues[$i]['member_name']);
}

// 後処理
$shopping = null;

// テンプレートを用いて表示
$tpl->display('sp_confirm.tpl');
```

■sp_confirm.tpl

(c:¥xampp¥htdocs¥shopping¥template)

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta http-equiv="content-language" content="ja">
<meta charset="shift_jis">
</head>
<body>
<form action="sp_thanks.php" method="post">
商品：{$product_name} ({$product_price}円)<br>
購入者：{$member_name}<br>
<input type="submit" name="submit" value="完了">
<input hidden="hidden" name="product" value="{$product_code}">
<input hidden="hidden" name="member" value="{$member_code}">
</form>
</body>
</html>
```

■sp_thanks.php

(c:¥xampp¥htdocs¥shopping)

```
<?php
require_once('sp_smarty.php');
require_once('sp_model.php');

// cofirm.php で$_SESSION 変数にセットしたデータを変数に格納
$product_code = (isset($_POST['product'])) ? $_POST['product'] : '';
$member_code = (isset($_POST['member'])) ? $_POST['member'] : '';

// クラス生成
$shopping = new Shopping();

// $product_code が存在するか確認
$shopping->setProductCode($product_code);
$shopping->productcodeCheck();

// $member_code が存在するか確認
$shopping->setMemberCode($member_code);
$shopping->membercodeCheck();

// product テーブルのデータを取得
$shopping->setProductCode($product_code);
$arrValues = $shopping->productData();
for ($i = 0 ; $i < count($arrValues); $i++) {
    // $product_code に対応する商品データをテンプレートにセット
    $tpl->assign('product_code', $arrValues[$i]['product_code']);
    $tpl->assign('product_name', $arrValues[$i]['product_name']);
    $tpl->assign('product_price', $arrValues[$i]['product_price']);
}

// $member_code に対応する商品データを取得
$shopping->setMemberCode($member_code);
$arrValues = $shopping->memberData();
```

```
for ($i = 0 ; $i < count($arrValues); $i++) {
    // $member_code に対応する商品データをテンプレートにセット
    $tpl->assign('member_code', $arrValues[$i]['member_code']);
    $tpl->assign('member_name', $arrValues[$i]['member_name']);
}

// shopping_order テーブルの code 例の最大値+1 を取得
$shopping->shoppingorderData();
$next_code = $shopping->getNextCode();

// shopping_order テーブルにデータを登録
$shopping->setProductCode($product_code);
$shopping->setMemberCode($member_code);
$shopping->shoppingorderDataInsert();

// 登録した shopping_order テーブルのデータ取得
$arrValues = $shopping->shoppingorderData2($next_code);
$tpl->assign('shoppingdata', $arrValues);

// 後処理
$shopping = null;

// テンプレートを用いて表示
$tpl->display('sp_thanks.tpl');
```

■sp_thanks.tpl

(c:¥xampp¥htdocs¥shopping¥template)

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta http-equiv="content-language" content="ja">
<meta charset="shift_jis">
</head>
<body>
次の通り、購入手続きを行いました。<br>
  商品：{$product_name} ({$product_price}円)<br>
  購入者：{$member_name}<br>
<br>
<table border="1">
 <tr>
   <th>code</th>
   <th>member</th>
   <th>product</th>
   <th>order_date</th>
 </tr>
 <!--- {foreach from=$shoppingdata item=var} --->
 <tr>
   <td>{$var.shopping_code}</td>
   <td>{$var.shopping_member}</td>
   <td>{$var.shopping_product}</td>
   <td>{$var.shppping_order_date}</td>
 </tr>
 <!--- {/foreach} --->
</table>
<br>
<a href="sp_entry.php">sp_entry.php</a>
</body>
</html>
```