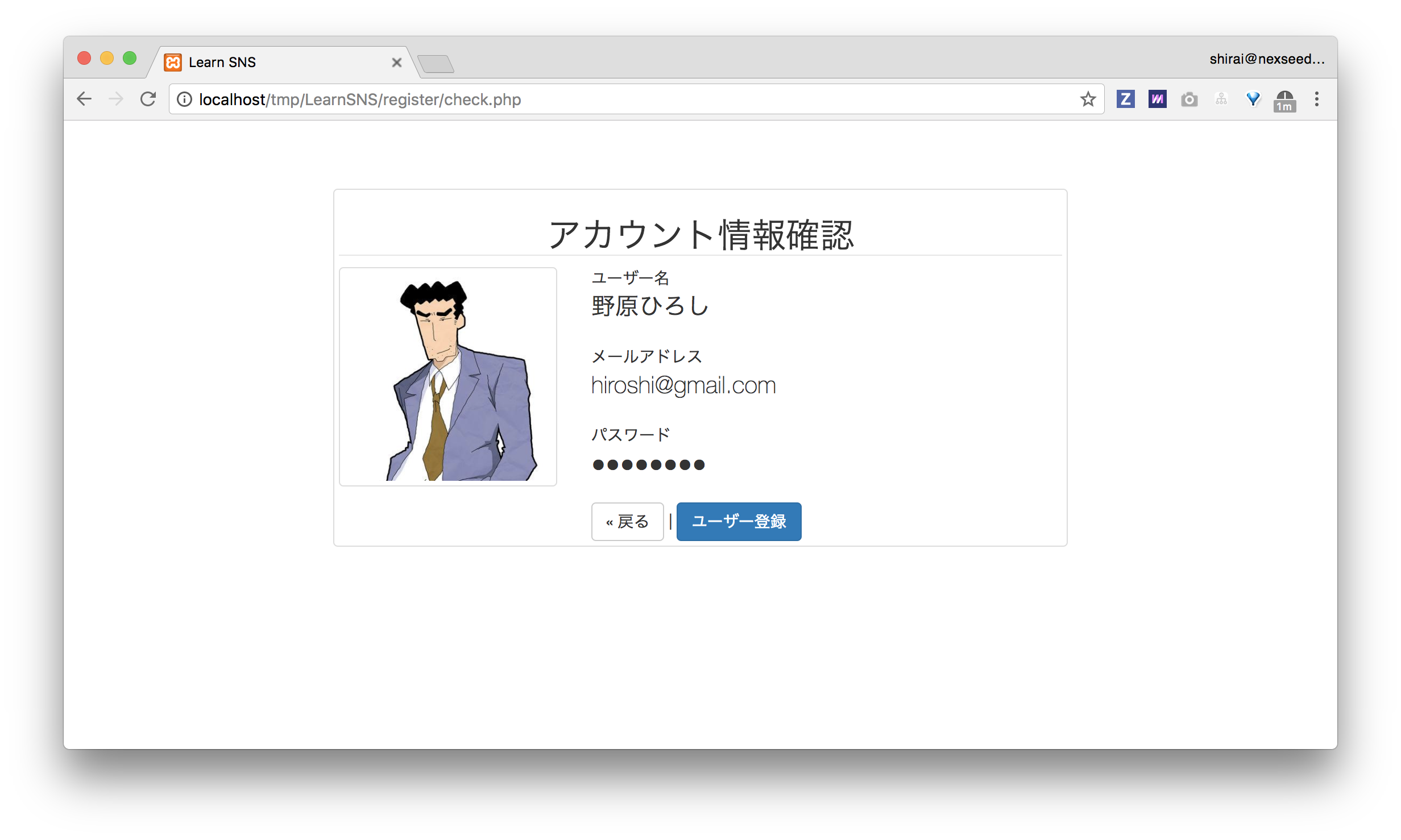
# **SNSシステム開発 - ユーザー登録機能**

ここでは、**ユーザー認証機能**のコアになる「データベースへのユーザー情報登録」をcheck.phpに実装していきます。

## **概要**

前回、check.phpでセッションに保存した送信データをテスト的に出力しました。

今回はsignup.phpから受け取った送信データをデータベースに登録する処理を中心に解説・実装していきます。

確認するのはcheck.phpですが、signup.phpから情報を入力し正しいフローでcheck.phpへ遷移する必要があるので注意が必要です。（直接check.phpへアクセスすることはできないように実装するため） 

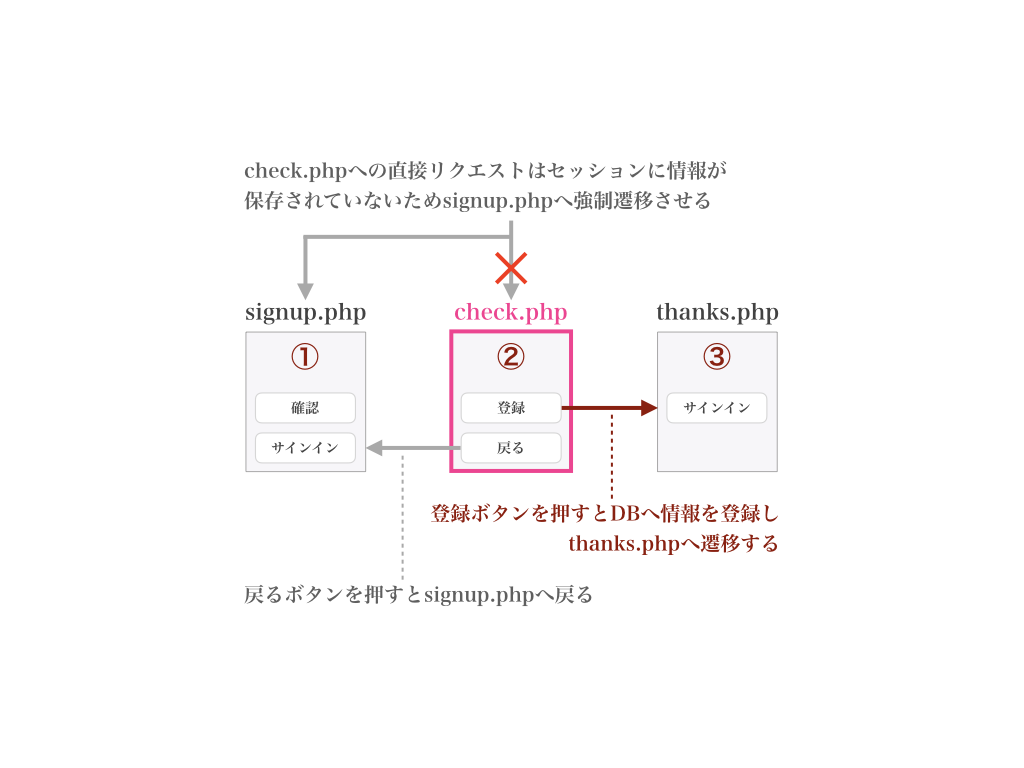
## **要件確認**

続いて、入力情報確認画面の要件を確認していきます。

### **要件リスト**

1. ユーザーは入力情報に間違いがないか確認することができる
2. 正規のルートを通らずにcheck.phpへのリクエストが行われた場合はsignup.phpへ強制遷移する
3. 入力情報に間違いがあった場合は戻るボタンを押すことでsignup.phpへ戻ることができる
4. ユーザー登録ボタンを押すことで入力情報をデータベースへ登録する
5. 登録処理完了後、アカウント作成完了ページ(thanks.php)へ遷移する

### **フローマップ**

****

## 

## **入力情報確認画面の表示を完成させる**

要件が確認できたところで、check.phpの表示内容を完成させましょう。

前回check.phpへ記述した$\_SESSIONを使用した出力プログラムは削除し、HTML側に埋め込む形に変更していきます。

##### **check.php**

**<?php**  
 **session\_start();**  
  
 **// ①**  
 **$name = $\_SESSION['LearnSNS']['name'];**  
 **$email = $\_SESSION['LearnSNS']['email'];**  
 **$password = $\_SESSION['LearnSNS']['password'];**  
 **$img\_name = $\_SESSION['LearnSNS']['img\_name'];**  
**?>**  
**<!DOCTYPE html>**  
<html lang**=**"ja">  
<head>  
 <meta charset**=**"utf-8">  
 <title>Learn SNS</title>  
 <link rel**=**"stylesheet" type**=**"text/css" href**=**"../assets/css/bootstrap.css">  
 <link rel**=**"stylesheet" type**=**"text/css" href**=**"../assets/font-awesome/css/font-awesome.css">  
 <link rel**=**"stylesheet" type**=**"text/css" href**=**"../assets/css/style.css">  
</head>  
<body style**=**"margin-top: 60px">  
 <div class**=**"container">  
 <div class**=**"row">  
 <div class**=**"col-xs-8 col-xs-offset-2 thumbnail">  
 <h2 class**=**"text-center content\_header">アカウント情報確認</h2>  
 <div class**=**"row">  
 <div class**=**"col-xs-4">  
 <img src**=**"../user\_profile\_img/<?php echo htmlspecialchars($img\_name); ?>" class**=**"img-responsive img-thumbnail">  
 </div>  
 <div class**=**"col-xs-8">  
 <div>  
 <span>ユーザー名</span>  
 <p class**=**"lead">**<?php echo htmlspecialchars($name); ?>**</p>  
 </div>  
 <div>  
 <span>メールアドレス</span>  
 <p class**=**"lead">**<?php echo htmlspecialchars($email); ?>**</p>  
 </div>  
 <div>  
 <span>パスワード</span>  
 *<!-- ② -->*  
 <p class**=**"lead">●●●●●●●●</p>  
 </div>  
 *<!-- ③ -->*  
 <form method**=**"POST" action**=**"">  
 *<!-- ④ -->*  
 <a href**=**"signup.php?action=rewrite" class**=**"btn btn-default">&laquo;&nbsp;戻る</a> |   
 *<!-- ⑤ -->*  
 <input type**=**"hidden" name**=**"action" value**=**"submit">  
 <input type**=**"submit" class**=**"btn btn-primary" value**=**"ユーザー登録">  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <script src**=**"../assets/js/jquery-3.1.1.js"></script>  
 <script src**=**"../assets/js/jquery-migrate-1.4.1.js"></script>  
 <script src**=**"../assets/js/bootstrap.js"></script>  
</body>  
</html>

各ポイントになる部分の上にコメントアウトで番号が振られているので、下記で解説を加えます。

### **①$\_SESSIONをわかりやすい名前で変数化**

$\_SESSIONのままだと読みづらく、タイポの原因にもなるので、それぞれわかりやすい変数名で変数化しておきます。以降この変数を使用して画面へ値の出力を行います。

### **②パスワードの表示**

パスワードはセキュリティ上画面に出力しないのがWEBサービス開発での鉄則となるため、単に「●」などを記述して非表示としておきます。

#### **※htmlspecialchars()関数を使ったXSS攻撃対策**

パスワード以外の情報は変数をechoする形で出力していますが、すべての変数にhtmlspecialchars()関数を使用しています。この関数は**XSS(クロスサイトスクリプティング)**という攻撃を防ぐための関数で、ユーザーが直接フォームなどで入力した情報を画面に出力する場合などに変数に対して使用します。

この対策を行わなかった場合、signup.php側で情報を入力する際にHTMLタグを入力して送信することでcheck.php側でそのままプログラムとして処理されてしまい画面のデザインが崩れます。またこれはHTMLだけでなくJavaScriptなどのプログラムも同様に処理されてしまうため、対策を怠るとサービス自体を破壊されたりユーザー情報をデータベースから抜かれてしまうことも。

対策方法さえ知っていれば問題ないので詳細まで理解しておく必要はありませんが、例題などが載っているサイトリンクを下記に記載します。気になる方は確認、またはググってみてください。

[XSS攻撃への対策](http://qiita.com/mpyw/items/565b3670dd0c7f9162fa)

### **③formについて**

今回もformタグを使ったPOST送信を自分自身へ送信する形で情報を処理していきます。

### **④戻るボタン (aタグ) について**

こちらは後ほど実装していく戻る処理に必要なボタン(aタグ)を画面に生成しています。詳細はこのあと解説していきますが、hrefに指定しているURLの末尾にaction=rewriteというパラメータを追加した状態でsignup.phpへリクエストを送信している点が重要になってきます。

### **⑤input type="hidden"について**

こちらはデータベースへデータへデータ登録する際に必要になるもので、こちらも詳細はのちほど解説します。

## **正規リクエストチェック（signup.phpへの強制遷移）**

それではまずはじめに、要件リストの中の正規リクエストチェックを行っていきます。フローマップの上部に記載された部分で、check.phpへの直接リクエストがあった場合は強制的にsignup.phpへ遷移するようプログラムします。

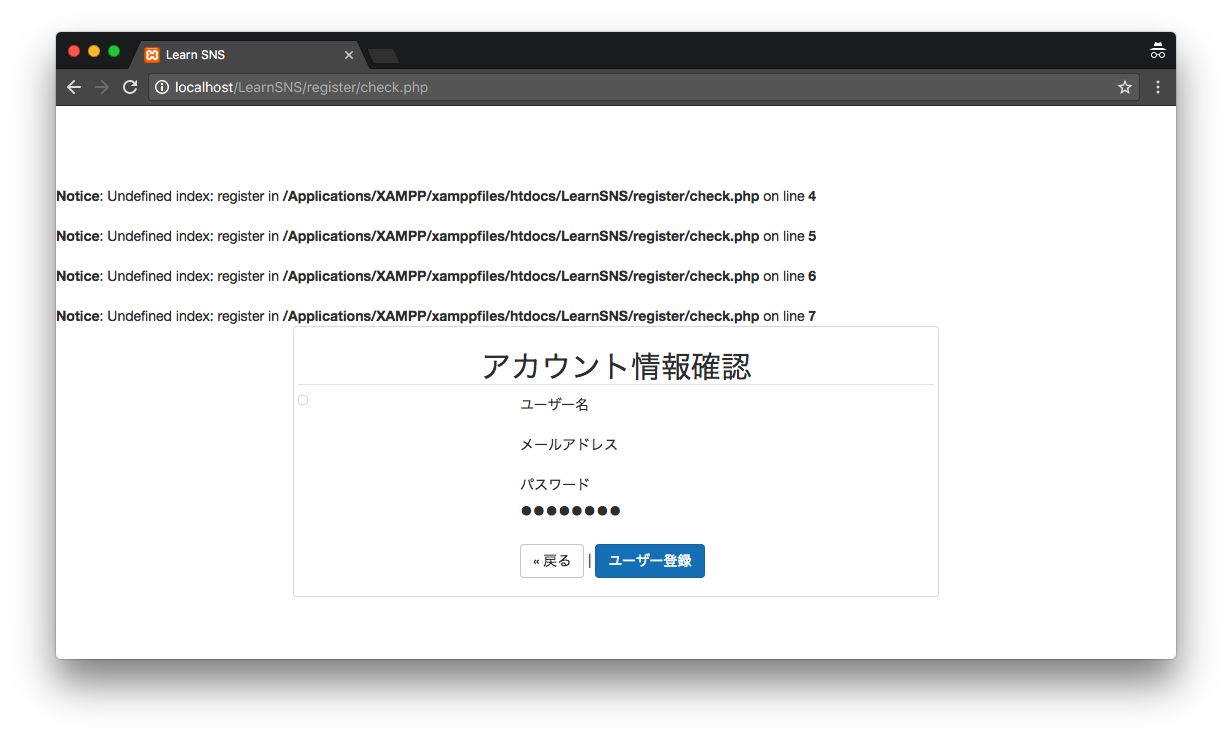
なぜそうする必要があるかというと、直接アクセスがあった場合はセッションに入力情報がそもそも保存されていないため$\_SESSIONを使用している箇所すべてにエラーがでてしまうためです。

試しにGoogle Chromeのシークレットブラウザモードを使用してエラーがでることを確かめてみましょう。

シークレットブラウザモードとは、セッションなどの情報をすべて破棄したまっさらな状態でWEBページを閲覧することができるモードで、cmd + shift + nキーでウィンドウを表示できます。（Windowsの方はcmdをctlへ読み替えて入力）

すると黒っぽいデザインでシークレットブラウザが開くので、localhost/LearnSNS/register/check.phpへアクセスしてみてください。

下記のようにエラーがでます。



これを回避するための対策として、もしセッションに必要な情報が保存されていなければsignup.phpへ遷移させ、情報を入力して正規のルートでcheck.phpへ来ることを促します。

##### **check.php**

**<?php**  
 session\_start();  
  
 **if** (**!**isset($\_SESSION['LearnSNS'])) {  
 header('Location: signup.php');  
 **exit**();  
 }  
  
 ・・・  
**?>**  
<!DOCTYPE html>  
<html lang="ja">  
・・・

この記述により、もし$\_SESSION['LearnSNS']が存在しなければ（!isset()で変数が存在しなかった場合trueを返す）header()関数を使用してsignup.phpへ強制的に遷移するようになりました。

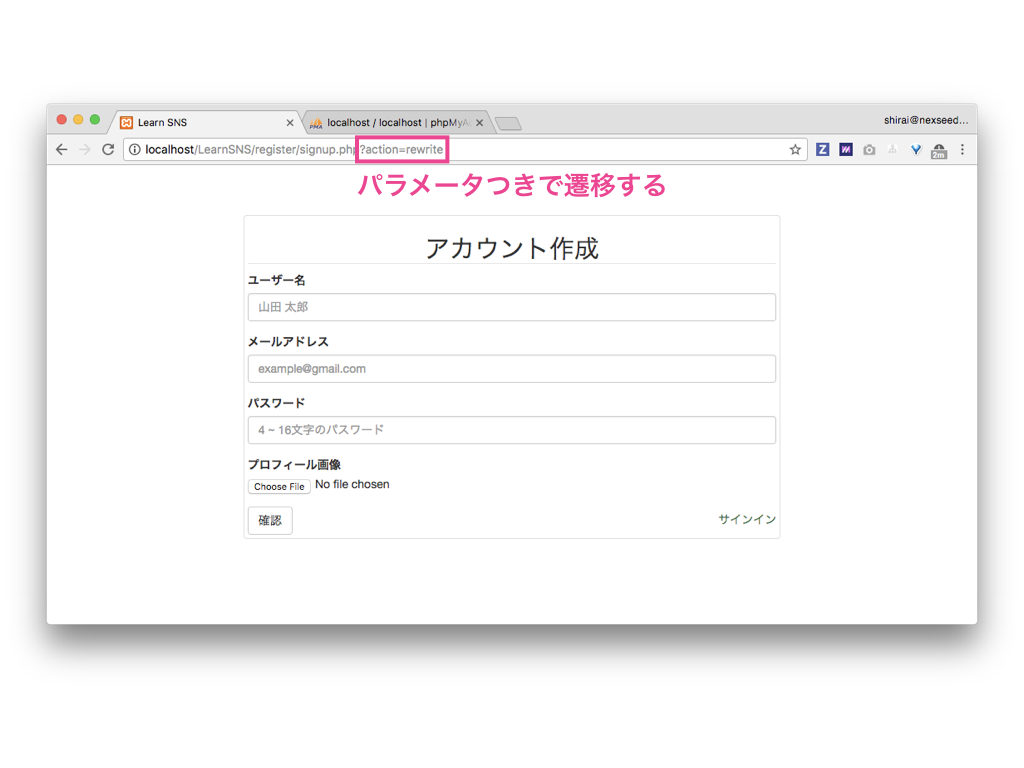
再度シークレットブラウザモードで開いているcheck.phpをリロードすると、signup.phpが表示されます。

これでsignup.phpから正規ルートでのアクセスでなければcheck.phpを確認することができないようになりました。

## **戻るボタンの実装**

上記④で少し取り上げた戻るボタンの実装を完成させます。

この戻るボタンはaタグで作成しているので、ボタンを押した際はPOST送信などのデータ送信はせず普通にhrefで指定するsignup.phpへ遷移します。ただこの時、パラメータとしてaction=rewriteが与えられた状態で遷移するようにhrefのパスを記述していますので、実際にこのボタンを押してsignup.phpへ遷移した際URLが下記の状態になります。



このパラメータの存在を利用して、戻るボタンを押してsignup.phpを表示した際の処理をプログラムしていきます。

現在、signup.phpへは戻るボタンで戻ってきても元々入力していた情報は消えてしまっており、ユーザーからするとすべて再入力するのは手間です。よくある実装機能ですが、戻るボタンを押してページへ戻ってきた場合は前回入力した項目はそのまま入力フォームに残っている状態にしましょう。

実装する必要がある処理は下記。

1. パラメータが存在すれば$\_POSTに$\_SESSIONの情報を代入
2. $file\_name = $\_FILES['input\_img\_name']['name']の部分をパラメータが存在する時処理しない
3. nameとemail部分はinputタグのvalue属性に変数をセット
4. パスワードは再入力を促すエラーメッセージを出力

### **1. パラメータが存在すれば$\_POSTに$\_SESSIONの情報を代入**

action=rewriteパラメータが存在するのはcheck.phpの戻るボタンを押して遷移してきた時のみなので、パラメータが存在する時 = セッションに前回の入力情報も存在します。

ここではセッションの情報から前回入力情報を使って$\_POSTを擬似的に作成し、!empty()のif文が実行されるようにします。また、すべての処理を行ってcheck.phpに遷移してしまわないように$errorsにrewriteをキーにした値を代入し$errorsが空にならないようにしておきます。

空のままだとempty($errors)を通過してしまいcheck.phpへ遷移してしまうため。また$errors['rewrite']に代入する値は何でも構わないのですが、今回はtrueを代入しておきます。

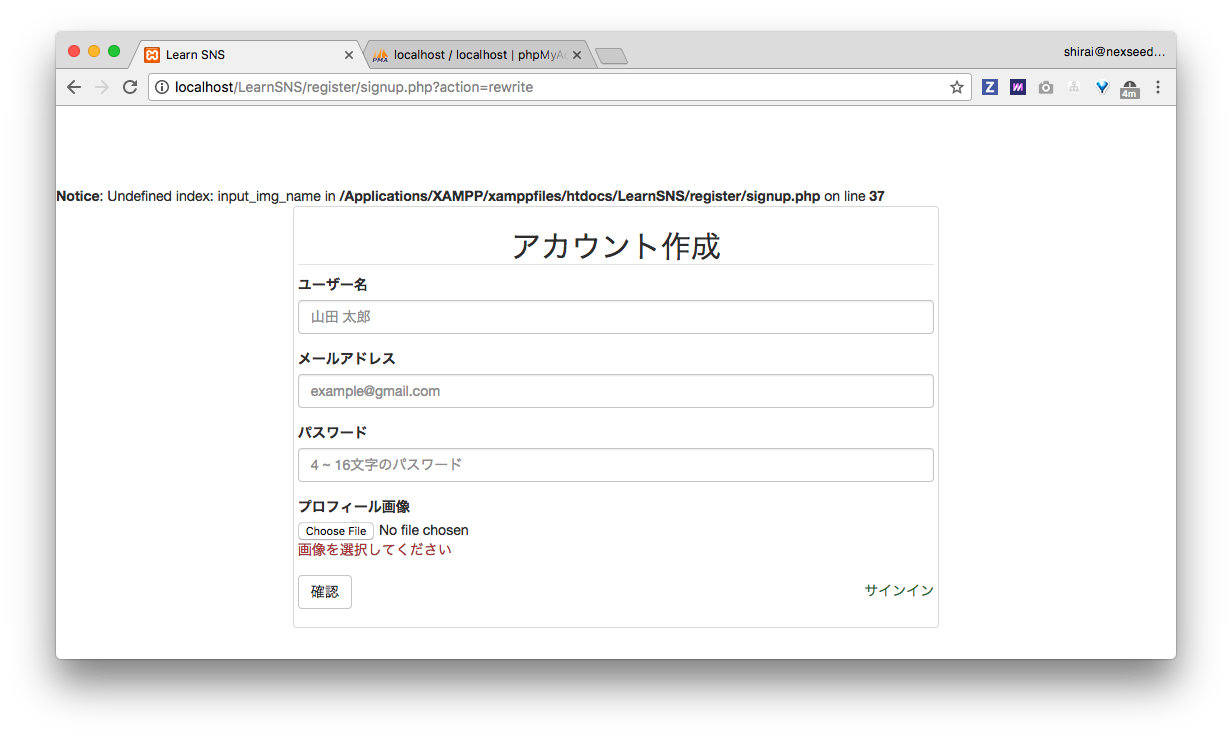
##### **signup.php**

**<?php**  
 session\_start();  
  
 **if** (isset($\_GET['action']) **&&** $\_GET['action'] **==** 'rewrite') {  
 $\_POST['input\_name'] **=** $\_SESSION['LearnSNS']['name'];  
 $\_POST['input\_email'] **=** $\_SESSION['LearnSNS']['email'];  
 $\_POST['input\_password'] **=** $\_SESSION['LearnSNS']['password'];  
  
 $errors['rewrite'] **=** **true**;  
 }  
  
・・・

### 

### **2. $file\_name = $\_FILES['input\_img\_name']['name']の部分をパラメータが存在する時処理しない**

上記プログラム記述後、signup.phpをリロードすると、下記のように$\_FILES['input\_img\_name']['name']のinput\_img\_nameが未定義だとエラーがでてしまいます。



$\_FILESのデータは再度input type="file"のフォームから選択して送信し直さなければデータが生成されないため、action=rewriteパラメータが存在するときは処理しないようにプログラムします。

##### **signup.php**

**<?php**  
 session\_start();  
 ・・・  
 *// パスワードの空チェック*  
 $count **=** strlen($password);  
 **if** ($password **==** '') {  
 $errors['password'] **=** 'blank';  
 } **elseif** ($count **<** 4 **||** 16 **<** $count) {  
 $errors['password'] **=** 'length';  
 }  
  
 *// 画像名を取得*  
 $file\_name **=** ''; *// ①*  
 **if** (**!**isset($\_GET['action'])) { *// ②*  
 $file\_name **=** $\_FILES['input\_img\_name']['name'];  
 }  
 **if** (**!empty**($file\_name)) {  
・・・

①の部分でまず空の$file\_nameを定義し、②でもしパラメータが存在しなければフォームからデータが送信されたことになるのでファイル名を$file\_nameに上書きする処理を記述しています。

これでsignup.phpをリロードするとエラーはなくなりました。

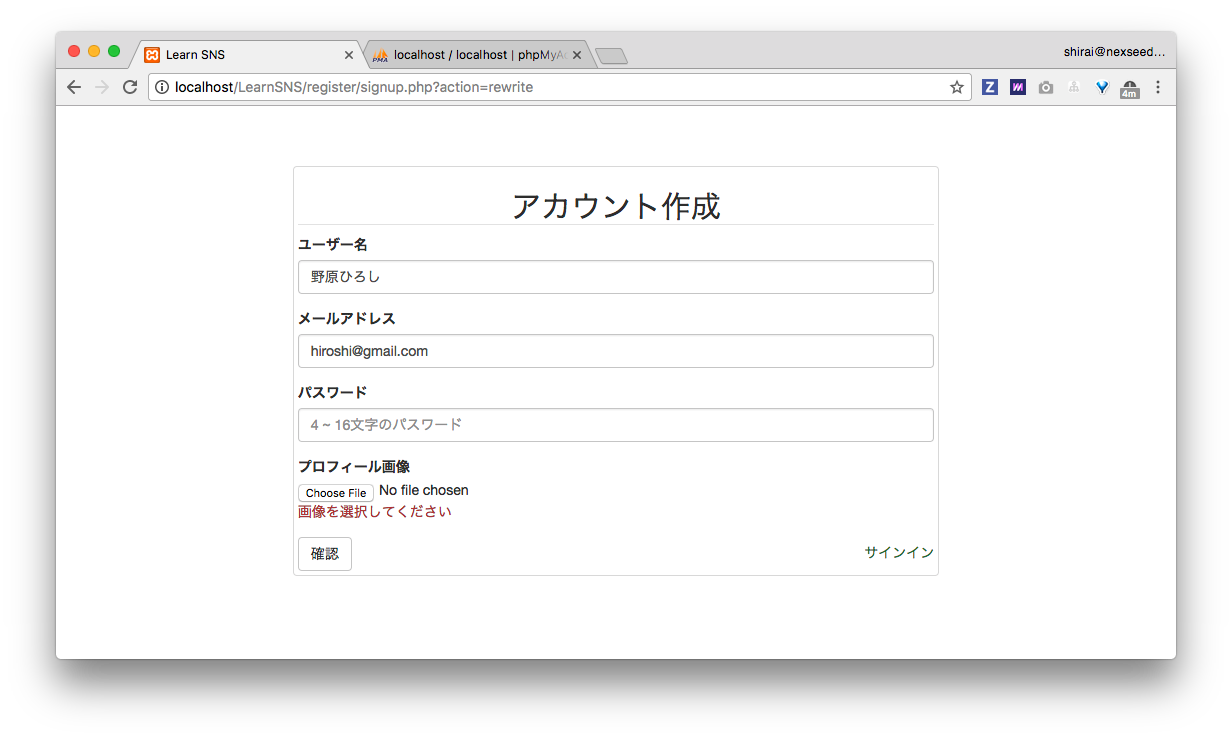
### **3. nameとemail部分はinputタグのvalue属性に変数をセット**

これでエラーなくsignup.phpに戻ってくることができたので、$\_POSTで受け取った前回入力情報を入力フォームに出力します。

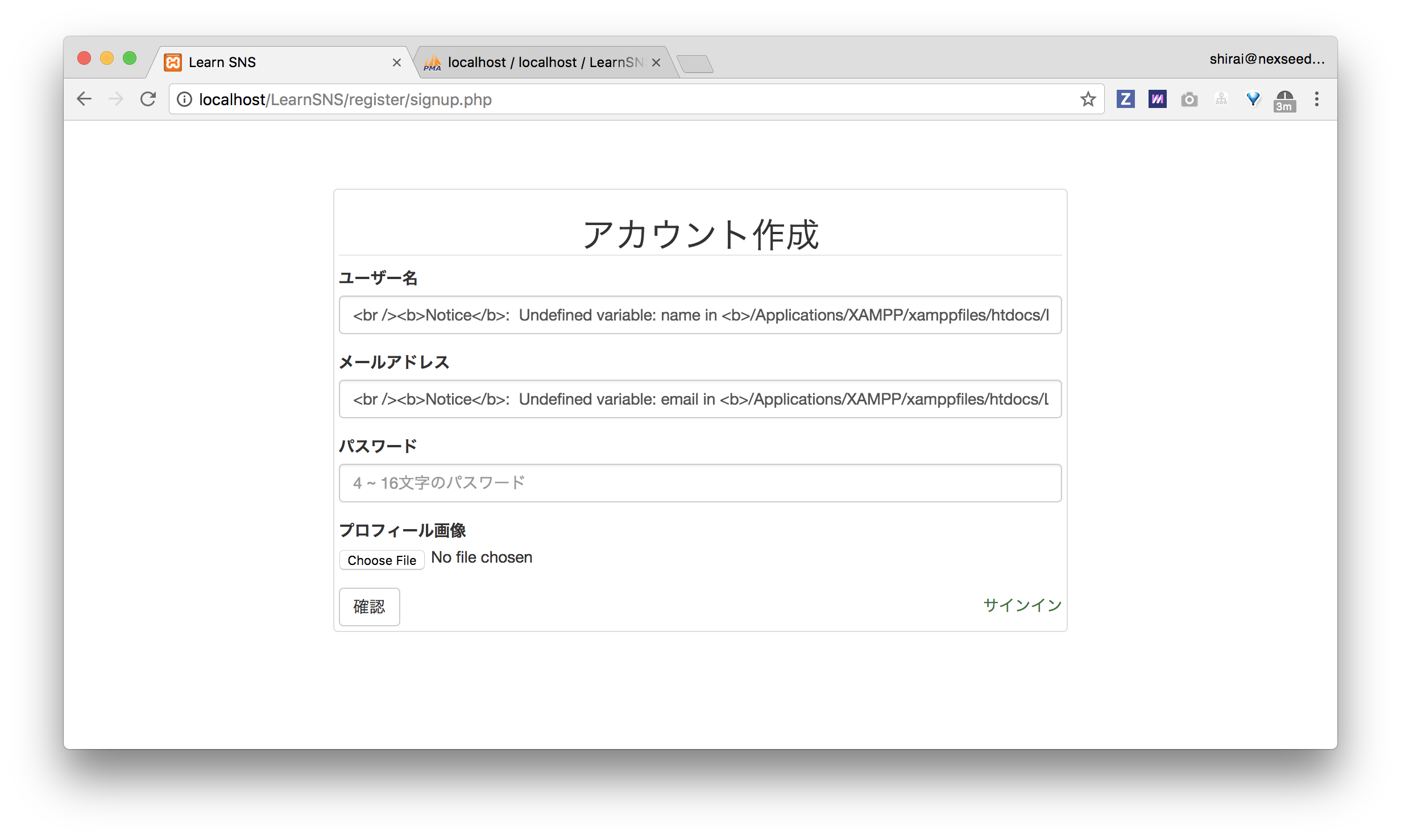
入力フォームの初期値として情報を出力する場合、value属性を使用します。

##### **signup.php** <form method**=**"POST" action**=**"signup.php" enctype**=**"multipart/form-data"> <div class**=**"form-group"> <label for**=**"name">ユーザー名</label> *<!-- inputタグのvalue属性の値に$nameを出力 -->* <input type**=**"text" name**=**"input\_name" class**=**"form-control" id**=**"name" placeholder**=**"山田 太郎" value**=**"<?php echo htmlspecialchars($name); ?>"> ・・・ </div> <div class**=**"form-group"> <label for**=**"email">メールアドレス</label> *<!-- inputタグのvalue属性の値に$nameを出力 -->* <input type**=**"email" name**=**"input\_email" class**=**"form-control" id**=**"email" placeholder**=**"example@gmail.com" value**=**"<?php echo htmlspecialchars($email); ?>"> ・・・ </div>

このプログラムにより、下記のようにsignup.phpに戻ってきた際の初期値として前回入力した情報が出力されます。



ただ、この場合URLパラメータが存在しないと下記のようなエラーが出てしまいます。



これは、$nameと$emailの変数が存在していないのにinput value属性に使用しているため。このエラーを解消するために、if文の上に空文字を代入したそれぞれの変数を定義しておきます。

##### **signup.php**

**<?php**  
 session\_start();  
 $errors **=** **[]**;  
 **if** (isset($\_REQUEST['action']) **&&** $\_REQUEST['action'] **==** 'rewrite') {  
 $\_POST['input\_name'] **=** $\_SESSION['LearnSNS']['name'];  
 $\_POST['input\_email'] **=** $\_SESSION['LearnSNS']['email'];  
 $\_POST['input\_password'] **=** $\_SESSION['LearnSNS']['password'];  
  
 $errors['rewrite'] **=** **true**;  
 }  
  
 *// 空で変数を定義*  
 $name **=** '';  
 $email **=** '';  
・・・

### **4. パスワードは再入力を促すエラーメッセージを出力**

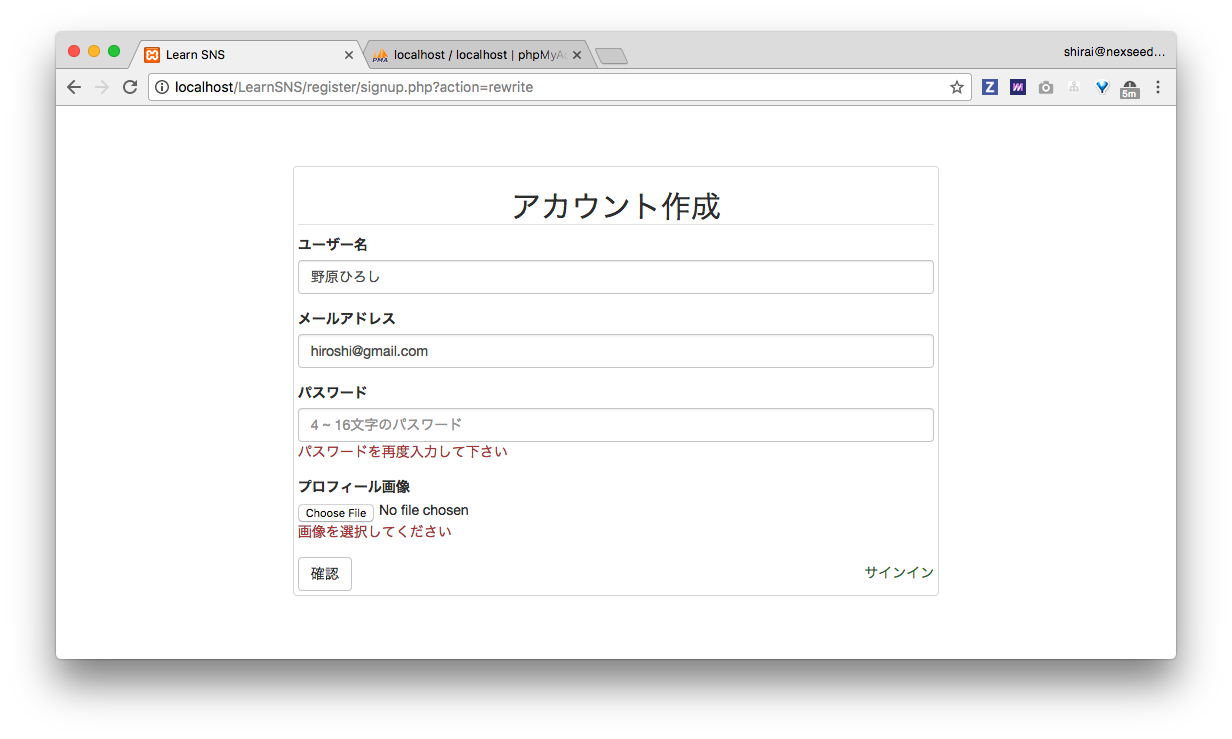
最後に、パスワードの入力フォームに再入力を促すエラーメッセージを出力します。パスワードはセキュリティ上戻る機能実装時には再入力を求めるほうが一般的です。

実装自体は画像フォームと同じで、もし$errorsが空じゃなければエラーメッセージを出力するようにプログラムします。

##### **signup.php**

・・・  
<div class**=**"form-group">  
 <label for**=**"password">パスワード</label>  
 <input type**=**"password" name**=**"input\_password" class**=**"form-control" id**=**"password" placeholder**=**"4 ~ 16文字のパスワード">  
 **<?php if(isset($errors['password']) && $errors['password'] == 'blank') { ?>**  
 <p class**=**"text-danger">パスワードを入力してください</p>  
 **<?php } ?>**  
 **<?php if(isset($errors['password']) && $errors['password'] == 'length') { ?>**  
 <p class**=**"text-danger">パスワードは4 ~ 16文字で入力してください</p>  
 **<?php } ?>**  
 *<!-- もし$errorsが空じゃなければエラーメッセージを出力する -->*  
 **<?php if(!empty($errors)) { ?>**  
 <p class**=**"text-danger">パスワードを再度入力して下さい</p>  
 **<?php } ?>**  
</div>  
・・・

ここまでの実装で、下記のように戻ってきた際の画面が完成しました。

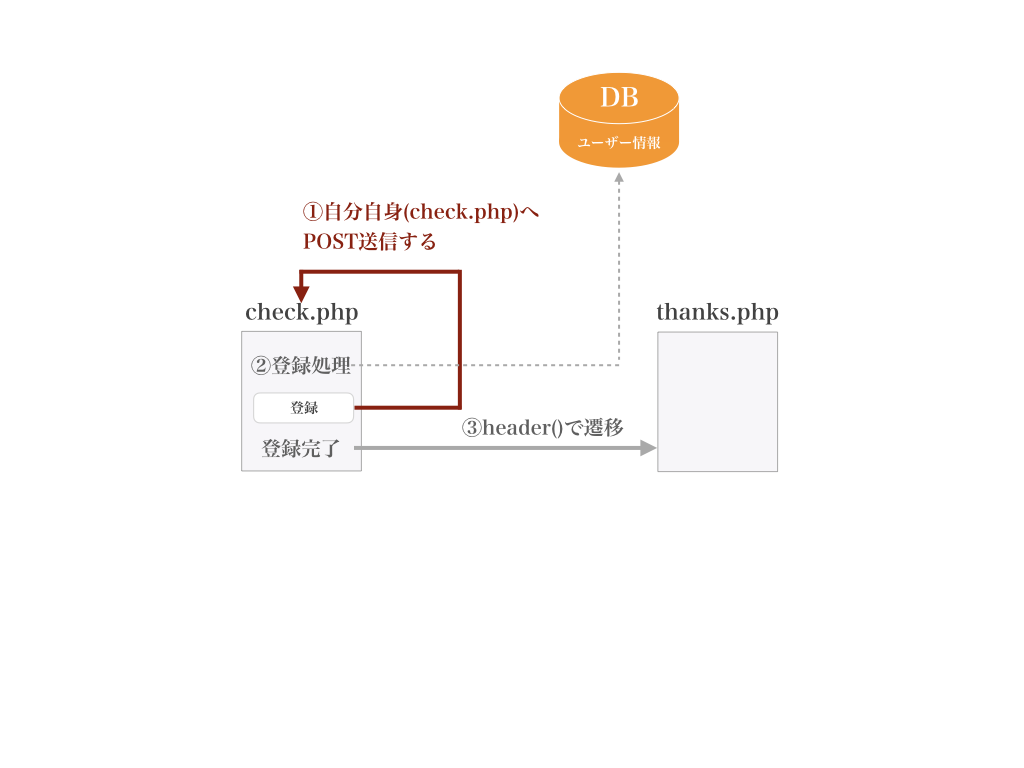


## **データベースへユーザー情報の登録**

戻るボタン機能の実装が済んだので、いよいよこのページの肝であるユーザー情報をデータベースへ登録する機能の実装に移りましょう。先ほどsignup.phpへ戻ったところから再度パスワードと画像を入力し、check.phpへ遷移したところから始めます。

処理の流れとしては下記の通り。

1. chek.phpへPOST送信する
2. POST送信があった時のみ処理するif文を記述する
   * PDOクラスとINSERT文を使用しデータベースへ登録する
3. 登録完了後はheader()関数でthanks.phpへ遷移する



ではひとつずつ進めていきます。

### **1. chek.phpへPOST送信する**

この部分に対応したformタグは既に記述済みなので、解説だけ加えます。

ユーザーがsignup.phpで入力した値は既にセッションに入っているので、POST送信で再度送る必要はありません。次でデータベースへ登録する際も$\_SESSIONの情報から生成された変数たちを使用します。

ただ、だからと言ってformタグの中に何もデータを入れないと、POST送信したところで$\_POSTは空のままになってしまい、if(!empty($\_POST))と空じゃなければ処理をするというプログラムが上手く動作しないため、今回はHTML記述時に⑤を振っていた箇所、<input type="hidden" name="action" value="submit">を記述しています。

name属性に指定しているキーや値になるvalueの中身はデータさえ入っていれば!empty()がtrueを返すため何でも良いので、actionとsubmitというキーと値をセットしています。

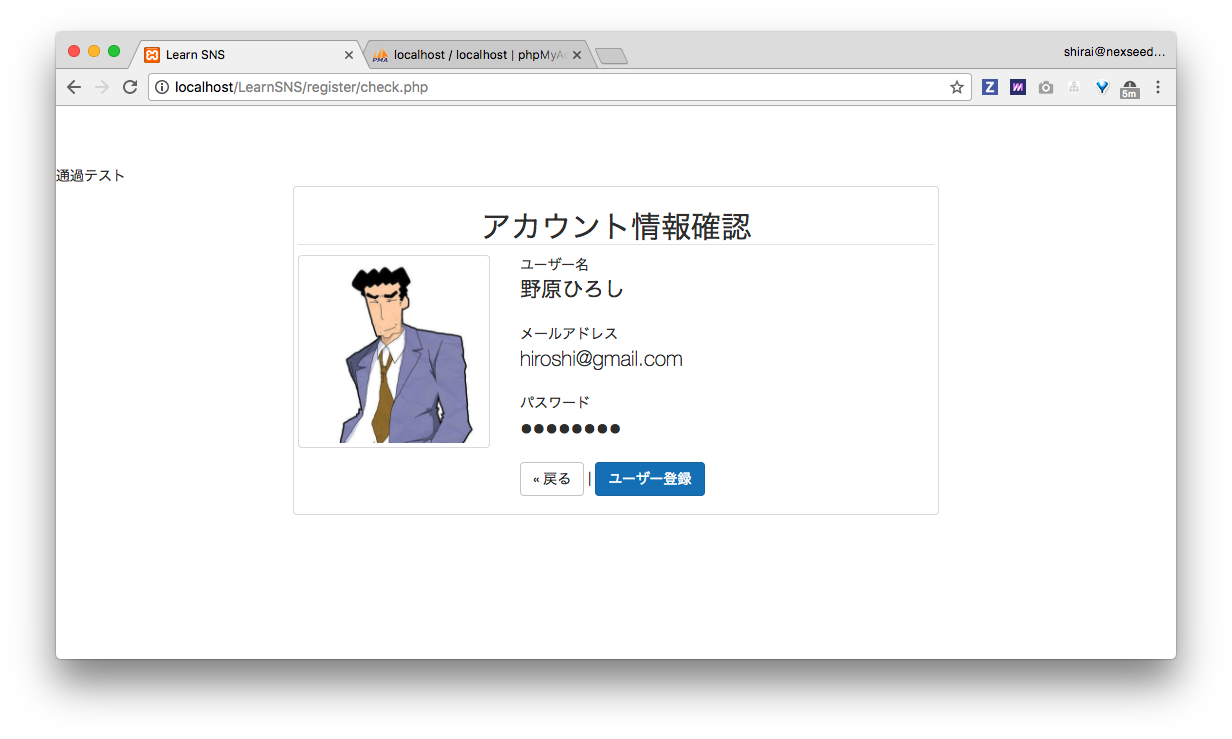
### **2. POST送信があった時のみ処理するif文を記述する**

さて、上記inputタグの設定で実際にフォーム送信時に$\_POSTが生成されるので、それをボタンが押されたと判定して処理するプログラムを記述します。

##### **check.php**

**<?php**  
 session\_start();  
  
 **if** (**!**isset($\_SESSION['LearnSNS'])) {  
 header('Location: signup.php');  
 **exit**();  
 }  
  
 $name **=** $\_SESSION['LearnSNS']['name'];  
 $email **=** $\_SESSION['LearnSNS']['email'];  
 $password **=** $\_SESSION['LearnSNS']['password'];  
 $img\_name **=** $\_SESSION['LearnSNS']['img\_name'];  
  
 *// 登録ボタンが押された時のみ処理するif文*  
 **if** (**!empty**($\_POST)) {  
 *// この中のデータベース登録処理を記述します*  
 **echo** '通過テスト' **.** '<br>';  
 }  
  
**?>**  
・・・

登録ボタンを押した際にこのif文を通過したかのチェックを行うテストecho文を記述したので、登録ボタンを押して左上部分に「通過テスト」と出力されるのを確認しましょう。

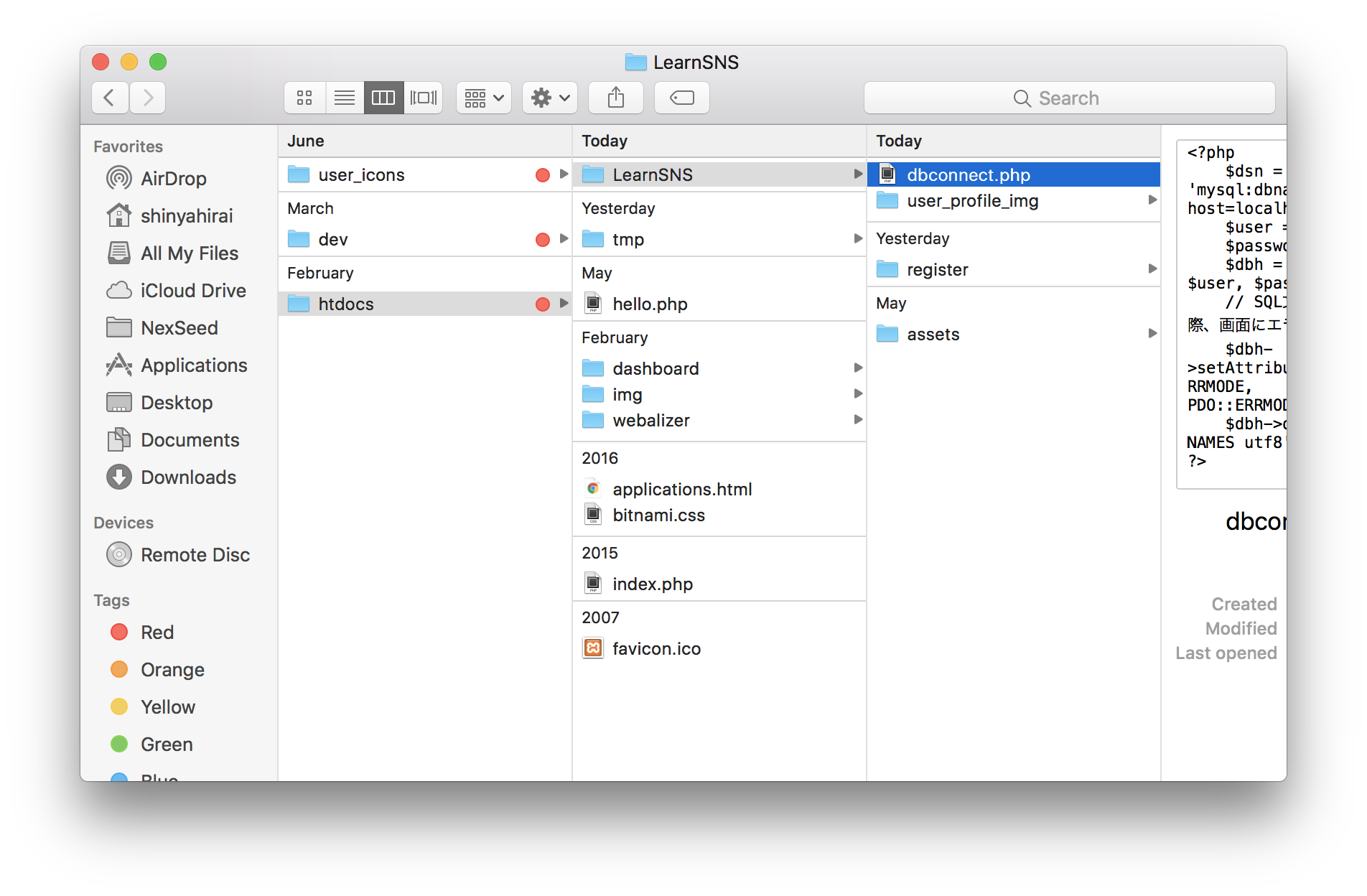


#### **PDOクラスとINSERT文を使用しデータベースへ登録する**

##### **データベース接続プログラム**

ユーザー情報をデータベースへ登録するために、データベース接続を行うプログラムを記述します。ただ今回は、他にもさまざまなページで同じデータベース接続プログラムを記述する必要があるため、ひとつのファイルにまとめて接続プログラムを記述しそのファイルをそれぞれのファイルから読み出すように構成します。

まず、dbconnect.phpファイルをLearnSNSフォルダ内に作成し下記プログラムをデータベース接続プログラムを記述します。

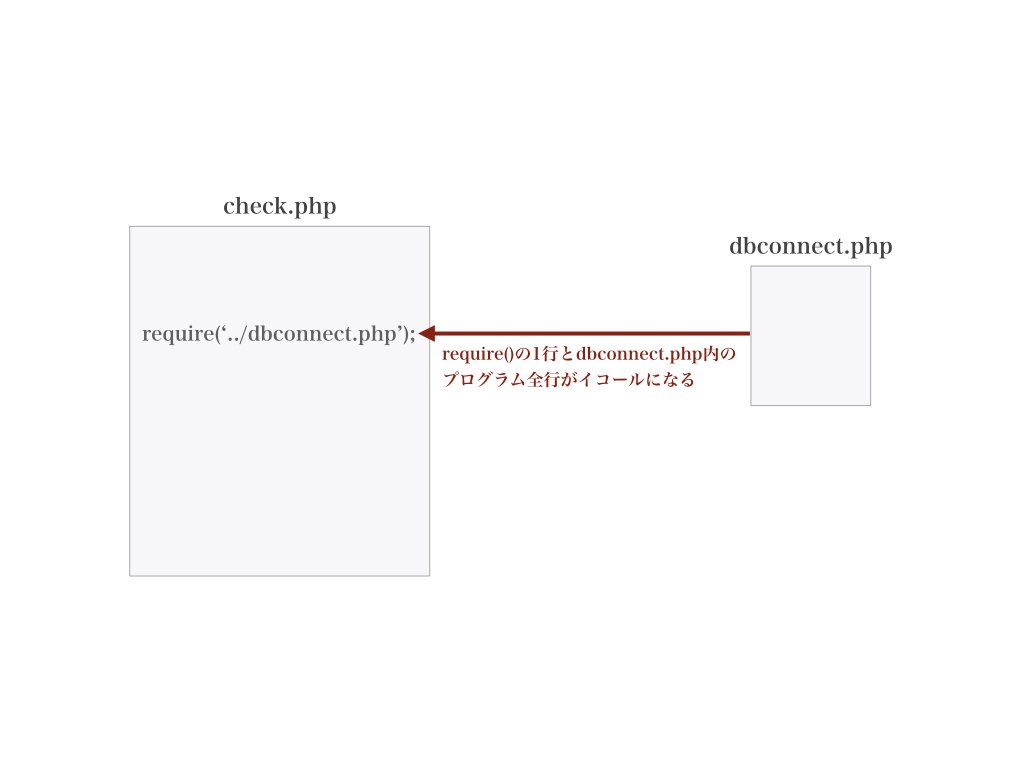


##### **dbconnect.php**

**<?php**  
 $dsn **=** 'mysql:dbname=LearnSNS;host=localhost';  
 $user **=** 'root';  
 $password**=**'';  
 $dbh **=** **new** PDO($dsn, $user, $password);  
 *// SQL文にエラーがあった際、画面にエラーを出力する設定*  
 $dbh**->**setAttribute(PDO**::**ATTR\_ERRMODE, PDO**::**ERRMODE\_EXCEPTION);  
 $dbh**->**query('SET NAMES utf8');  
**?>**

作成したdbconnect.phpをcheck.phpに読み込む処理を記述します。

別ファイルを読み込む処理にはrequire()文を使用します。require()文は()内に読み込みたいファイルのパスを記述するだけでrequire()を記述した行に読込ファイル内のプログラムをすべて置き換えます。



##### **check.php**

**<?php**  
 session\_start();  
 **require**('../dbconnect.php');  
・・・

##### **登録処理**

これでデータベースとの接続は完了したので、実際の登録処理に入っていきます。

上記で記述したif文の中にデータベースへ登録するプログラムを記述します。（テストecho文は削除します）

#### **check.php**

**<?php**  
 session\_start();  
 **require**('../dbconnect.php');  
 ・・・  
 $name **=** $\_SESSION['LearnSNS']['name'];  
 $email **=** $\_SESSION['LearnSNS']['email'];  
 $password **=** $\_SESSION['LearnSNS']['password'];  
 $img\_name **=** $\_SESSION['LearnSNS']['img\_name'];  
  
 *// 登録ボタンが押された時のみ処理するif文*  
 **if** (**!empty**($\_POST)) {  
 $sql **=** 'INSERT INTO `users` (`name`, `email`, `password`, `img\_name`, `created`) VALUES (?,?,?,?, NOW())';  
 $data **=** **[**$name, $email, password\_hash($password, PASSWORD\_DEFAULT), $img\_name];  
 $stmt **=** $dbh**->**prepare($sql);  
 $stmt**->**execute($data);  
 }  
  
**?>**  
・・・

PDOを使ったSQL文の実行方法やINSERT文については別ユニットで既に解説済みのためここでは省きます。

新しく取り上げるものとしてはpassword\_hash()関数について。

##### **パスワードのハッシュ化**

パスワードはデータベースに登録する際、ハッシュ化という文字列の暗号化のような処理を必ず行います。

そのままの文字列を登録してしまうとセキュリティ上よくありません。例えば、もし悪意ある技術者がデータベースに不正アクセスした場合でも、パスワードをハッシュ化し元のパスワードがわからない状態にしておけば最低限の被害におさえることができるためです。

// ハッシュ化の例  
 hogehoge → $2y$10$XdbJN1gMyCAoB77oDO2.fOZ1PcnPa2x105v/fkJlJ3raC3aR2LsCG

PHPではハッシュ化のために用意された関数がいくつか存在しますが、現時点でもっとも安全性の高いpassword\_hash()関数を今回は使用します。

password\_hash()関数は、第一引数にハッシュ化したい文字列を、第二引数にPASSWORD\_DEFAULTを指定して使用します。

**<?php**  
 $password **=** "hogehoge";  
 $hash\_password **=** password\_hash($password, PASSWORD\_DEFAULT);  
 **echo** $hash\_password;  
 *// 下記出力結果*  
 *// → $2y$10$XdbJN1gMyCAoB77oDO2.fOZ1PcnPa2x105v/fkJlJ3raC3aR2LsCG*  
**?>**

また、ハッシュ化したパスワードと別で入力された文字列とを比較する際はpassword\_verify()関数を使用しますが、こちらはサインイン機能の実装時に解説します。

パスワードのハッシュ化 / password\_hash()関数の詳しい説明は下記サイトが参考になるので、興味のある方は一読また自身でググってみてください。

[安全なハッシュ関数とpassword\_hash()](https://column.prime-strategy.co.jp/archives/column_1014)

[ハッシュ関数について/password\_hash()を利用する](http://noumenon-th.net/programming/2016/02/08/password_hash/)

### **4. 登録完了後はheader()関数でthanks.phpへ遷移する**

最後は登録完了画面であるthanks.phpへheader()関数を使って遷移します。

#### **check.php**

**<?php**  
 session\_start();  
 **require**('../dbconnect.php');  
 ・・・  
 $name **=** $\_SESSION['LearnSNS']['name'];  
 $email **=** $\_SESSION['LearnSNS']['email'];  
 $password **=** $\_SESSION['LearnSNS']['password'];  
 $img\_name **=** $\_SESSION['LearnSNS']['img\_name'];  
  
 *// 登録ボタンが押された時のみ処理するif文*  
 **if** (**!empty**($\_POST)) {  
 $sql **=** 'INSERT INTO `users` (`name`, `email`, `password`, `img\_name`, `created`) VALUES (?,?,?,?, NOW());  
 $data **=** [$name, $email, password\_hash($password, PASSWORD\_DEFAULT), $img\_name];  
 $stmt **=** $dbh**->**prepare($sql);  
 $stmt**->**execute($data);  
  
 unset($\_SESSION['LearnSNS']);  
 header('Location: thanks.php');  
 **exit**();  
 }  
  
**?>**  
・・・

この時、unset()文を使って$\_SESSION['LearnSNS']を削除します。unset()文は指定した変数もしくは配列を破棄することができます。

$\_SESSIONに保存した内容は別で不具合を起こさないよう不要になったら削除するのが鉄則です。

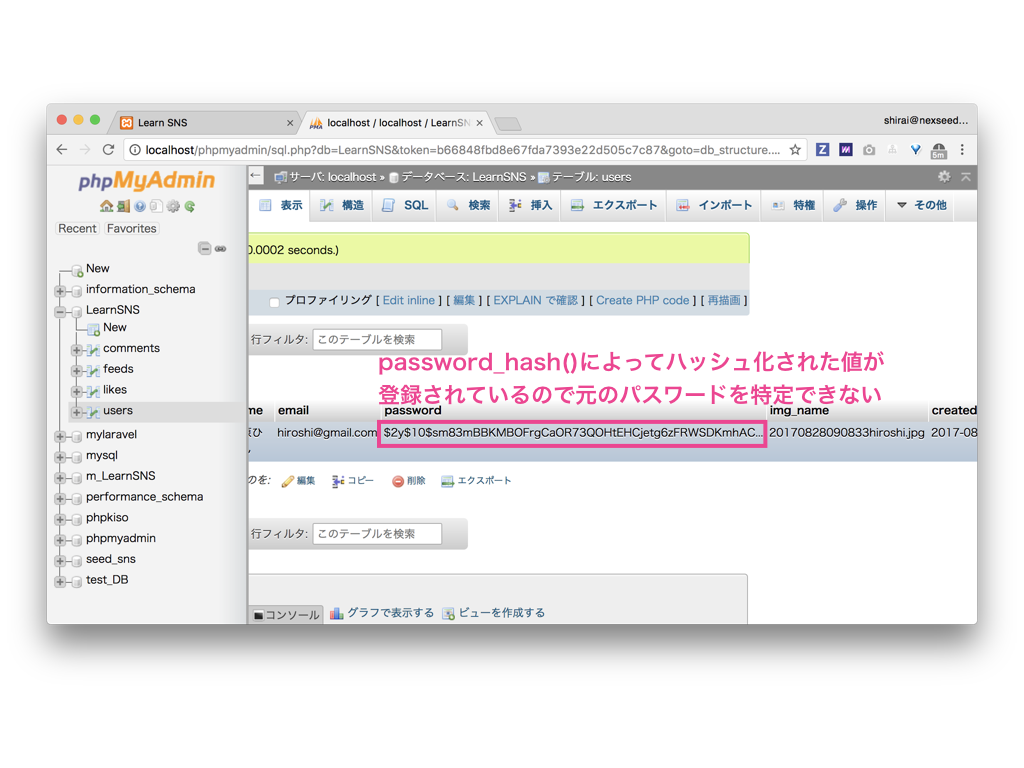
## **アカウント作成完了ページ(thanks.php)へ遷移する**

最後に、check.phpから「ユーザー登録ボタン」を押し、データベースへユーザー情報を登録してthanks.phpへ遷移することを確認します。

アカウント作成完了ページが表示されます。 

また、ブラウザでlocalhost/phpmyadminとPHPMyAdminを表示し、左のデータベースリストメニューからLearnSNSを選択し、usersテーブルをクリックして開きます。

そこに先ほどのユーザー情報が登録されていればデータベース登録処理も問題なく動作しています。 

また、passwordカラムに登録されたデータはpassword\_hash()関数によってハッシュ化された値が登録されていることが確認できます。 

# 

# 

# 

# **まとめ**

以上でcheck.phpの実装が完了しました。

### **実装した機能まとめ**

* 正規ルートでのアクセスチェック
* 戻るボタンの処理
* ユーザー情報の登録処理
* アカウント作成完了ページへ遷移

# 