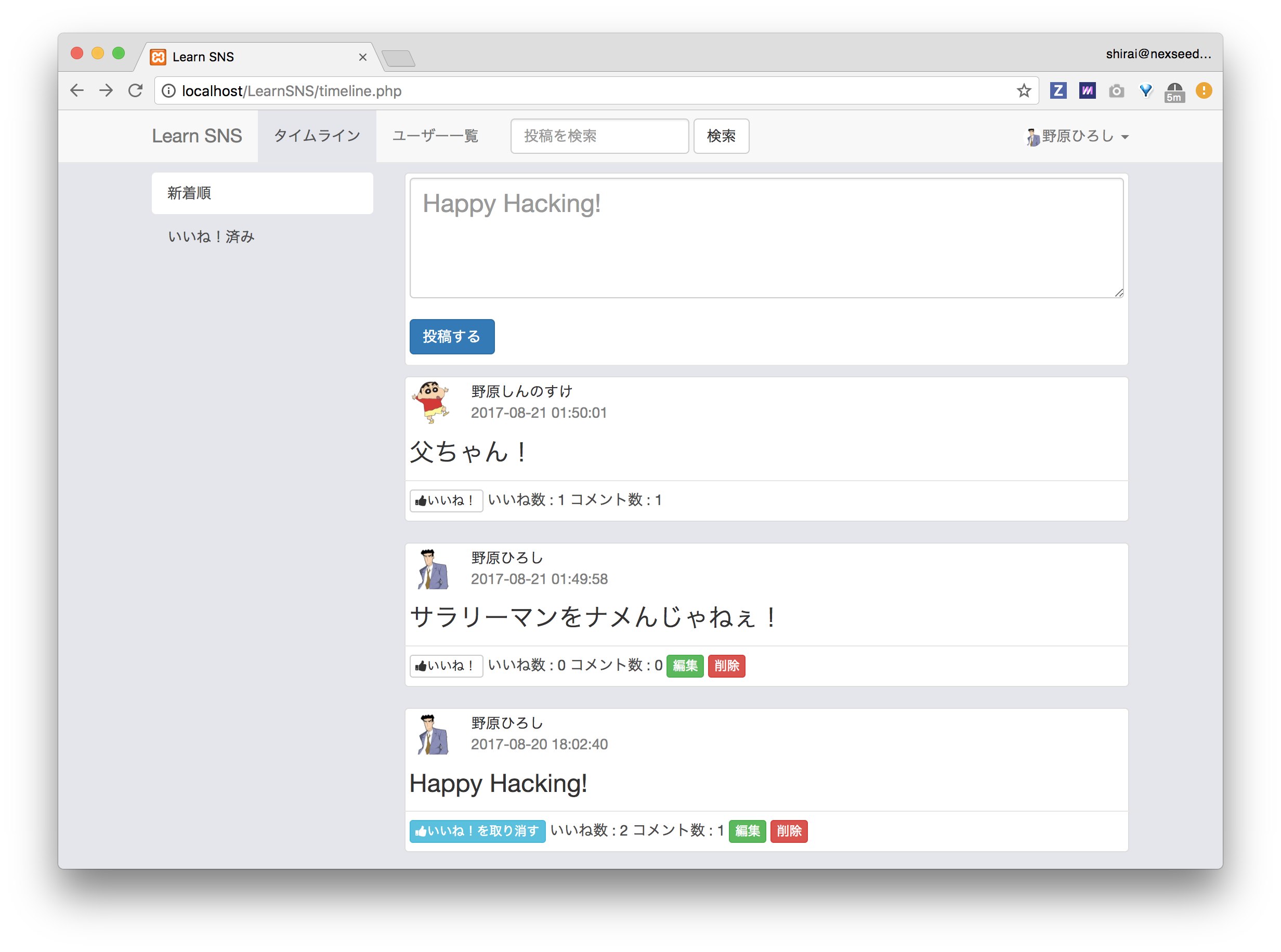
# **SNSシステム開発 - 編集・削除**

この項では、**編集・削除**の実装を行なっていきます。

## **概要**

このページでは、タイムライン画面のうちの以下の項目を題材に学習していきます。

1. 投稿の編集
2. 投稿の削除



## **要件確認**

続いて、タイムライン画面の要件を確認していきます。

### **要件リスト**

1. ログインしているユーザ名と画像が確認できる
2. ログアウトできる
3. 投稿ができる
   * 空チェックのバリデーションがある
4. ユーザーは投稿の全てを見ることができる
   * 各投稿には投稿したユーザ名と画像が表示されている
5. ユーザは投稿にいいねできる
6. **編集画面へ移動できる**
   * **編集ボタンから詳細画面へ遷移する**
7. **投稿の削除ができる**
8. ページネーションできる
9. いいねした記事だけにフィルタができる

## **編集・削除ボタンの表示・非表示**

要件が確認できたので、プログラミングに入っていきます。

現在、一覧画面には全てのユーザーの投稿が表示されるようになっていますが、編集ボタン、削除ボタンに関しても全ての投稿に表示されています。

本来、編集や削除は自分の投稿にのみ行えるはずですので、そもそもボタンが表示されていることがバグとなります。自分の投稿以外は編集、削除ボタンが表示されないように制御を加えていきましょう。

自分の投稿かどうかは、各投稿を行なったユーザーのIDと、現在ログインしているユーザーのIDが等しいかどうかで判別することができます。

これまでの実装で、投稿を行なったユーザーIDは$feed['user\_id']、現在ログインしているユーザーのIDは$signin\_user['id']で取得することができますね。

それでは実装していきましょう。

**timeline.php**

<?php if ($feed['user\_id'] == $signin\_user['id']): ?>

<a href="edit.php?feed\_id=<?php echo $feed['id'] ?>" class="btn btn-success btn-xs">編集</a>

<a onclick="return confirm('ほんとに消すの？');" href="delete.php?feed\_id=<?php echo $feed['id'] ?>" class="btn btn-danger btn-xs">削除</a>

<?php endif; ?>

先ほど説明した通り、各投稿を行なったユーザーのIDと、現在ログインしているユーザーのIDが等しいかで判別できるので、これをそのままif文の条件に指定しました。条件が満たされた時にだけ、つまり、自分の投稿にだけ、編集ボタンと削除ボタンが表示されます。

上記コードを書いた後に一度ページをリロードして確認してみましょう。

次にedit.phpとdelete.phpを作成します。

## **削除機能の作成**

まずは削除機能から実装していきます。

タイムライン画面ですでに削除確認ダイアログが実装されているので、このdelete.phpでは画面を用意する必要がありません。純粋に削除の機能のみを実装し、削除の処理が終了したらtimeline.phpに遷移させましょう。

**削除処理の流れ**

1. どの投稿を削除するか特定する
2. CRUD処理のDelete処理を実装する
3. timeline.phpに遷移する

LearnSNSで削除の処理を行うのは今回が初めてとなりますが、これまで習ってきた内容で十分応用が効く内容です。

下記に解答ソースコードがありますが、まずは考え、実践してみましょう。

**timeline.php**

href="delete.php?feed\_id=<?php echo $feed['id'] ?>"

まず最初に削除処理へ飛ばすリンクの修正が必要です。

今回はaタグでGET送信を利用しているのでGET送信のパラメータを利用して、どの投稿を削除するか特定できるようにしましょう。

GETでパラメータ送信する方法はURL?キー=値で指定します。

**delete.php**

<?php

require('dbconnect.php');

$feed\_id = $\_GET["feed\_id"];

$sql ="DELETE FROM `feeds` WHERE `id` = ?";

$data = [$feed\_id];

$stmt = $dbh->prepare($sql);

$stmt->execute($data);

header("Location: timeline.php");

exit();

削除処理に関しては、先程説明した通り、画面を要しません。**PHPの処理のみで解決する場合、PHPタグは開始タグのみ必要とし、終了タグは省略するのがマナー**となっています。

LearnSNSでDELETE文を利用するのは今回が初めてですが、DELETE文の構文さえ理解していれば、その他は既存の知識で実装できるはずです。

DELETE FROM テーブル名 WHERE id = 値;

## 

## 

## 

## 

## 

## **編集機能の作成**

続いて、編集機能を実装していきましょう。

**編集処理の流れ**

1. 編集する投稿の情報を取得
2. CRUD処理のUpdate処理を実装する
3. timeline.phpに遷移する

こちらも削除処理と同様に、編集の処理を行うのは今回が初めてですが、内容としてはこれまでの応用となります。是非解説を見る前に実装方法を自身で考えて挑戦してみましょう。

なお、編集画面でもnavbar.phpをincludeしています。サインインユーザーの情報が欠落しているため、編集画面でエラーが出てしまいます。タイムライン画面で実装した内容を編集画面にも組み込みましょう。（[007\_タイムライン機能その1](#_hcvpybm5h14c)）

また、削除処理と同様、どの投稿を編集するのか特定する必要があるので、こちらもリンクを修正しておきましょう。

**timeline.php**

href="edit.php?feed\_id=<?php echo $feed['id'] ?>"

**編集画面に表示するデータ取得**

編集画面に表示するデータですが、タイムライン画面で取得していた投稿情報一覧の中から選択された一つの情報に絞り込むだけです。

実行するSQL文は下記のようになるでしょう。

SELECT `feeds`.\*,`users`.`name`,`users`.`img\_name` FROM `feeds` LEFT JOIN `users` ON `feeds`.`user\_id`=`users`.`id` WHERE `feeds`.`id`= ?;

WHERE句で指定しているfeedsテーブルのidは、タイムライン画面からGETパラメータで渡されていますので、こちらを利用しましょう。

**edit.php**

<?php

require('dbconnect.php');

$feed\_id = $\_GET['feed\_id'];

$sql = 'SELECT `feeds`.\*,`users`.`name`,`users`.`img\_name` FROM `feeds` LEFT JOIN `users` ON `feeds`.`user\_id`=`users`.`id` WHERE `feeds`.`id`= ?';

$data = [$feed\_id];

$stmt = $dbh->prepare($sql);

$stmt->execute($data);

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

?>

**編集画面にデータを表示**

**edit.php**

<form class="form-group" method="post" action="edit.php">

<img src="user\_profile\_img/<?php echo $feed['img\_name']; ?>" width="60">

<?php echo $feed['name']; ?><br>

<?php echo $feed['created']; ?><br>

<textarea name="feed" class="form-control"><?php echo $feed['feed']; ?></textarea>

<input type="hidden" name="feed\_id" value="<?php echo $feed['id']; ?>" >

<input type="submit" value="更新" class="btn btn-warning btn-xs">

</form>

なお、編集後のtimeline.phpへの遷移はPHPで制御を行うので、formタグのaction属性の値はedit.phpに変更をしましょう。

**編集機能**

formタグの実装まで終わりましたので、最後に編集機能を実装していきます。

**edit.php**

if (!empty($\_POST)) {

$sql = 'UPDATE `feeds` SET `feed` = ? WHERE `id` = ?';

$data = [$\_POST['feed'], $\_POST['feed\_id']];

$stmt = $dbh->prepare($sql);

$stmt->execute($data);

header('Location: timeline.php');

exit();

}

編集処理が行われるのはformから送信が行われた時のみですので、いつものPOST送信が空かどうかのチェックが必要です。

SQL処理に関しては説明は不要でしょう。

これまで通り、実行したいSQL文を定義し、タプル処理を行い、実行するだけです。

**GETとPOST**

編集処理は実装できましたが、ソースコードに不備があります。

今のままでは、POST送信時にもGET送信時の処理が実行されてしまいます。

GET送信時、POST送信時、それぞれ行われる処理が切り替わるように条件を加えます。

<?php

require('dbconnect.php');

if (isset($\_GET['feed\_id']) {

$feed\_id = $\_GET['feed\_id'];

$sql = 'SELECT `feeds`.\*,`users`.`name`,`users`.`img\_name` FROM `feeds` LEFT JOIN `users` ON `feeds`.`user\_id`=`users`.`id` WHERE `feeds`.`id`= ?';

$data = [$feed\_id];

$stmt = $dbh->prepare($sql);

$stmt->execute($data);

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

}

if (!empty($\_POST)) {

$sql = 'UPDATE `feeds` SET `feed` = ? WHERE `id` = ?';

$data = [$\_POST['feed'], $\_POST['feed\_id']];

$stmt = $dbh->prepare($sql);

$stmt->execute($data);

header('Location: timeline.php');

exit();

}

?>

以上で編集、削除機能は完成です。

ここまでで、PHP+MySQL+HTML/CSSでCRUD処理を用いたアプリケーション作成の基礎が終わりました。

ここからはより実践的な内容となっていきます。

当然、応用が必要となってきますので、ここまでの内容はしっかりと復習しておきましょう。