# **SNSシステム開発 - ページネーション**

**タイムライン**の開発のつづきで、ページネーション機能を実装していきます。

## **概要**

このページでは、タイムラインの作成を題材に以下について学習していきます。

1. ページネーション機能

ページネーションとは、一覧表示を行うページによくある「次へ」「前へ」などです。

## **要件確認**

ページネーションの要件は以下の通りです。

* 1ページあたりの表示件数は最大5件
* 1ページ目の場合はNewerをクリックしても画面遷移しない
* 最終ページの場合はOlderをクリックしても画面遷移しない

実装する上で追加で必要な情報は以下2つのみです。

1. 1ページあたりの表示件数
2. 開いているページが何ページ目か

## **ページネーション実装**

1. **1ページあたりの表示件数を5件にする**

まず、表示件数を5件に変更します。

変更箇所は投稿情報を取得しているSQL文になります。取得件数を制限するLIMIT句というものを用いて表示件数を5件に設定しましょう。まずは下記のSQLを試してみましょう。

SELECT \* FROM `feeds` ORDER BY `created` LIMIT 5;

新しい投稿から5件だけが取得されたはずです。

上記SQL文のように、

LIMIT 制限したい数

と記述することで、任意の数に制限することができます。

なお、この制限は上限です。つまり、その数までなら取得することになるので、LIMIT 5 を指定した際にそもそも3件しかレコードがない場合、その3件が取得されます。

それでは、timeline.phpのSQLを改修しましょう。

$sql = 'SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id` ORDER BY `created` DESC LIMIT 5';

ここまで実装できたらブラウザで確認してみてください。タイムライン画面で投稿が5件のみ表示されていれば正解です。

1. **ページ毎に表示されるデータを変更する。**

今の状態だとDBから取得できるデータは毎回同じ5件です。1ページ目なら最初の5件、2ページ目なら次の5件、といったように、ページ毎に表示する内容を変える必要がありますね。まずは下記のSQLによって表示されるデータが変わることを確認しましょう。

SELECT \* FROM `feeds` ORDER BY `created` LIMIT 5 OFFSET 0;

SELECT \* FROM `feeds` ORDER BY `created` LIMIT 5 OFFSET 5;

一つ目のSQLと二つ目のSQLで取得された結果が変わったはずです。

OFFSET スキップするレコード数

OFFSET句は取得開始位置を調整することができるSQL文です。指定する数字はスキップするレコード数を示すので、一つ目のSQL文はOFFSET 0、つまり0スコードスキップするので、結局一番最初から取得することになります。それに対して、二つ目のSQL文はOFFSET 5、つまり5レコードスキップするので、6件目から取得することになります。

すでにLIMITを用いて5件取得するという要件は満たしているので、何件レコードをスキップするのか動的に算出できればページの切り替えが実装できますね。

1. **何ページ目を開いているか取得する**

取得するデータの数を制限する方法と、指定した箇所からデータを取得する方法がわかったので、次はPHPで動的に何ページ目を開いているかを変更できるようにします。

##### **timeline.php**

(PHP部分)

if (isset($\_GET['page'])) {

$page = $\_GET['page'];

} else {

$page = 1;

}

(HTML部分)

<ul class="pager">

<li class="previous">

<a href="timeline.php?page=<?php echo $page - 1; ?>">

<span aria-hidden="true">&larr; Newer</span>

</a>

</li>

<li class="next">

<a href="timeline.php?page=<?php echo $page + 1; ?>">

<span aria-hidden="true">Older &rarr;</span>

</a>

</li>

</ul>

ページ数の取得にはGETパラメータを利用します。GET送信時のパラメータはURL?キー=値と書きます。

今回はtimeline.php?page=値と書いていますが、timeline.phpの何ページ、という風に読んでもらえれば理解しやすいかと思います。

なお、liタグに付いているdisabledはここで外しておきましょう。

上記コードを書いた後にブラウザでOlder, Newerをクリックして画面を遷移すると、パラメータにpage=○という形で、何ページ目を開いているか確認できますね。

※なお、現時点では存在しないページにも移動できます。

1. **取得するデータをページ毎に変更する**

##### PHPの実装に入る前に、前提知識として必要となる、定数と、組み込み関数、max()、min()、ceil()について見ておきましょう。

##### **定数**

これまで変数を利用してきましたが、その類似として定数というものがあります。

変数が「変わりうる値」であったのに対して、定数は「定まった値」です。

つまり、一度宣言をするとそれ以降その値は変わることがないということです。

利用方法は変数と同じですが、宣言方法が異なります。

const 定数名 = 値;

なお、定数名は全て大文字で記載することが慣習となっています。

##### **max()**

max関数は引数の中でもっとも大きな値を算出します。

例えば、$num = max(1, 9);とすると、$numには9が代入されます。

##### **min()**

min関数はmax関数の逆で、もっとも小さい値を算出します。

例えば、$num = min(1, 9);とすると、$numには1が代入されます。

##### **ceil()**

ceil関数は端数の切り上げを行う関数で、引数の次に大きい**整数**を算出します。

例えば、$num = ceil(1.9);とすると、$numには2が代入されます。

四捨五入ではないので、$num = min(1.1);とした場合も、$numには2が代入されます。

今回必要となる前提知識は以上です。

それでは、実装を見ていきましょう。

##### **timeline.php**

const CONTENT\_PER\_PAGE = 5;

// -1などのページ数として不正な値を渡された場合の対策

$page = max($page, 1);

// ヒットしたレコードの数を取得するSQL

$sql\_count = "SELECT COUNT(\*) AS `cnt` FROM `feeds`";

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

$stmt\_count->execute();

$record\_cnt = $stmt\_count->fetch(PDO::FETCH\_ASSOC);

// 最後のページが何ページになるのか算出

// 最後のページ = 取得したページ数 ÷ 1ページあたりに表示する件数

$last\_page = ceil($record\_cnt['cnt'] / CONTENT\_PER\_PAGE);

// 最後のページより大きい値を渡された場合の対策

$page = min($page, $last\_page);

$start = ($page - 1) \* CONTENT\_PER\_PAGE;

$sql = 'SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id` ORDER BY `created` DESC LIMIT '. CONTENT\_PER\_PAGE .' OFFSET ' . $start;

組み込み関数に関しては、先ほど確認した通りです。実践でも使えるようにしましょう。

SQL文について、少し解説を加えておきます。

$sql\_count = "SELECT COUNT(\*) AS `cnt` FROM `feeds`";

SQLにもPHPと同様に組み込み関数が存在します。これまで使っていたNOW()などもその一つです。

今回利用しているCOUNT()も頻繁に使われる組み込み関数になります。

COUNT関数はレコードの件数を算出します。引数には任意の式を記述できますが、\*(アスタリスク)を利用することで、全レコードの数を求めることができます。

上記のSQL文ではfeedsテーブルの全レコード数が算出されます。

ASを利用しているのは、取得できたレコードのキーを分かりやすくするためです。

そのままでは、$record\_cnt['COUNT(\*)']と取り出すことになります。キーがSQLの組み込み関数というのは分かりにくい上に、何よりプログラマ的にナンセンスです。

ASを利用することで、ASで指定した値をキーとして利用することができるので、$record\_cnt['cnt']で取り出すことができます。

$sql = 'SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id` ORDER BY `created` DESC LIMIT '. CONTENT\_PER\_PAGE .' OFFSET ' . $start;

こちらのSQLは一見すると、長く複雑に思えますが、1と2で説明したLIMITとOFFSETが加わっただけで、ベースは結合を利用したSELECT文です。一語一語しっかりと意味を確認しながら読んで見てください。

1. **不正なページングを防ぐ**

今の状態でも正しくデータの取得はできますが、最後のページまでいってもOlderがクリックできたり、Newerは押せなさそうな表示になるのに実際には押せてしまいます。

最初のページでは前へ、最後のページでは次へ進めないように制御を加えましょう。

ページ切り替えボタンの箇所を以下のように変更します。

<?php if ($page == 1): ?>

<li class="previous disabled"><a><span aria-hidden="true">&larr;</span> Newer</a></li>

<?php else: ?>

<li class="previous"><a href="timeline.php?page=<?= $page - 1; ?>"><span aria-hidden="true">&larr;</span> Newer</a></li>

<?php endif; ?>

<?php if ($page == $last\_page): ?>

<li class="next disabled"><a>Older <span aria-hidden="true">&rarr;</span></a></li>

<?php else: ?>

<li class="next"><a href="timeline.php?page=<?= $page + 1; ?>">Older <span aria-hidden="true">&rarr;</span></a></li>

<?php endif; ?>

制御が正しくできているかブラウザで試してみましょう。

NewerもOlderもどちらも流れは同じです。開いているページが、1ページ目ならNewerのリンクを無効にし、最終ページならOlderを無効にしています。

これで、ページネーション機能は完成です。