docs.zendframework.com

Adding zend-paginator to the Album Module

1. Docs »
2. MVC Tutorials »
3. Adding zend-paginator to the Album Module

In this tutorial, we will use the zend-paginator component to add a handy pagination controller to the bottom of the album list.

Currently, we only have a handful of albums to display, so showing everything on one page is not a problem. However, how will the album list look when we have 100 albums or more in our database? The standard solution to this problem is to split the data up into a number of pages, and allow the user to navigate around these pages using a pagination control. Just type "Zend Framework" into Google, and you can see their pagination control at the bottom of the page:

Preparation

As before, we are going to use sqlite, via PHP's PDO driver. Create a text file `data/album-fixtures.sql` with the following contents:

```
 INSERT INTO "album" ("artist", "title")
VALUES
    ("David Bowie", "The Next Day (Deluxe Version)"),
    ("Bastille", "Bad Blood"),
    ("Bruno Mars", "Unorthodox Jukebox"),
    ("Emeli Sandé", "Our Version of Events (Special Edition)"),
    ("Bon Jovi", "What About Now (Deluxe Version)"),
    ("Justin Timberlake", "The 20/20 Experience (Deluxe Version)"),
    ("Bastille", "Bad Blood (The Extended Cut)"),
```

```
    ("P!nk", "The Truth About Love"),
    ("Sound City - Real to Reel", "Sound City - Real to Reel"),
    ("Jake Bugg", "Jake Bugg"),
    ("Various Artists", "The Trevor Nelson Collection"),
    ("David Bowie", "The Next Day"),
    ("Mumford & Sons", "Babel"),
    ("The Lumineers", "The Lumineers"),
    ("Various Artists", "Get Ur Freak On - R&B Anthems"),
    ("The 1975", "Music For Cars EP"),
    ("Various Artists", "Saturday Night Club Classics - Ministry of
Sound"),
    ("Hurts", "Exile (Deluxe)"),
    ("Various Artists", "Mixmag - The Greatest Dance Tracks of All
Time"),
    ("Ben Howard", "Every Kingdom"),
    ("Stereophonics", "Graffiti On the Train"),
    ("The Script", "
    ("Stornoway", "Tales from Terra Firma"),
    ("David Bowie", "Hunky Dory (Remastered)"),
    ("Worship Central", "Let It Be Known (Live)"),
    ("Ellie Goulding", "Halcyon"),
    ("Various Artists", "Dermot O'Leary Presents the Saturday Sessions
2013"),
    ("Stereophonics", "Graffiti On the Train (Deluxe Version)"),
    ("Dido", "Girl Who Got Away (Deluxe)"),
    ("Hurts", "Exile"),
    ("Bruno Mars", "Doo-Wops & Hooligans"),
    ("Calvin Harris", "18 Months"),
    ("Olly Murs", "Right Place Right Time"),
    ("Alt-J (?)", "An Awesome Wave"),
    ("One Direction", "Take Me Home"),
    ("Various Artists", "Pop Stars"),
    ("Various Artists", "Now That's What I Call Music! 83"),
    ("John Grant", "Pale Green Ghosts"),
    ("Paloma Faith", "Fall to Grace"),
    ("Laura Mvula", "Sing To the Moon (Deluxe)"),
    ("Duke Dumont", "Need U (100%) [feat. A*M*E] - EP"),
    ("Watsky", "Cardboard Castles"),
    ("Blondie", "Blondie: Greatest Hits"),
    ("Foals", "Holy Fire"),
    ("Maroon 5", "Overexposed"),
    ("Bastille", "Pompeii (Remixes) - EP"),
```

```
    ("Imagine Dragons", "Hear Me - EP"),
    ("Various Artists", "100 Hits: 80s Classics"),
    ("Various Artists", "Les Misérables (Highlights From the Motion
Picture Soundtrack)"),
    ("Mumford & Sons", "Sigh No More"),
    ("Frank Ocean", "Channel ORANGE"),
    ("Bon Jovi", "What About Now"),
    ("Various Artists", "BRIT Awards 2013"),
    ("Taylor Swift", "Red"),
    ("Fleetwood Mac", "Fleetwood Mac: Greatest Hits"),
    ("David Guetta", "Nothing But the Beat Ultimate"),
    ("Various Artists", "Clubbers Guide 2013 (Mixed By Danny Howard) -
Ministry of Sound"),
    ("David Bowie", "Best of Bowie"),
    ("Laura Mvula", "Sing To the Moon"),
    ("ADELE", "21"),
    ("Of Monsters and Men", "My Head Is an Animal"),
    ("Rihanna", "Unapologetic"),
    ("Various Artists", "BBC Radio 1's Live Lounge - 2012"),
    ("Avicii & Nicky Romero", "I Could Be the One (Avicii vs. Nicky
Romero)"),
    ("The Streets", "A Grand Don't Come for Free"),
    ("Tim McGraw", "Two Lanes of Freedom"),
    ("Foo Fighters", "Foo Fighters: Greatest Hits"),
    ("Various Artists", "Now That's What I Call Running!"),
    ("Swedish House Mafia", "Until Now"),
    ("The xx", "Coexist"),
    ("Five", "Five: Greatest Hits"),
    ("Jimi Hendrix", "People, Hell & Angels"),
    ("Biffy Clyro", "Opposites (Deluxe)"),
    ("The Smiths", "The Sound of the Smiths"),
    ("The Saturdays", "What About Us - EP"),
    ("Fleetwood Mac", "Rumours"),
    ("Various Artists", "The Big Reunion"),
    ("Various Artists", "Anthems 90s - Ministry of Sound"),
    ("The Vaccines", "Come of Age"),
    ("Nicole Scherzinger", "Boomerang (Remixes) - EP"),
    ("Bob Marley", "Legend (Bonus Track Version)"),
    ("Josh Groban", "All That Echoes"),
    ("Blue", "Best of Blue"),
    ("Ed Sheeran", "+"),
    ("Olly Murs", "In Case You Didn't Know (Deluxe Edition)"),
```

```
    ("Macklemore & Ryan Lewis", "The Heist (Deluxe Edition)"),
    ("Various Artists", "Defected Presents Most Rated Miami 2013"),
    ("Gorgon City", "Real EP"),
    ("Mumford & Sons", "Babel (Deluxe Version)"),
    ("Various Artists", "The Music of Nashville: Season 1, Vol. 1
(Original Soundtrack)"),
    ("Various Artists", "The Twilight Saga: Breaking Dawn, Pt. 2
(Original Motion Picture Soundtrack)"),
    ("Various Artists", "Mum - The Ultimate Mothers Day Collection"),
    ("One Direction", "Up All Night"),
    ("Bon Jovi", "Bon Jovi Greatest Hits"),
    ("Agnetha Fältskog", "A"),
    ("Fun.", "Some Nights"),
    ("Justin Bieber", "Believe Acoustic"),
    ("Atoms for Peace", "Amok"),
    ("Justin Timberlake", "Justified"),
    ("Passenger", "All the Little Lights"),
    ("Kodaline", "The High Hopes EP"),
    ("Lana Del Rey", "Born to Die"),
    ("JAY Z & Kanye West", "Watch the Throne (Deluxe Version)"),
    ("Biffy Clyro", "Opposites"),
    ("Various Artists", "Return of the 90s"),
    ("Gabrielle Aplin", "Please Don't Say You Love Me - EP"),
    ("Various Artists", "100 Hits - Driving Rock"),
    ("Jimi Hendrix", "Experience Hendrix - The Best of Jimi Hendrix"),
    ("Various Artists", "The Workout Mix 2013"),
    ("The 1975", "Sex"),
    ("Chase & Status", "No More Idols"),
    ("Rihanna", "Unapologetic (Deluxe Version)"),
    ("The Killers", "Battle Born"),
    ("Olly Murs", "Right Place Right Time (Deluxe Edition)"),
    ("A$AP Rocky", "LONG.LIVE.A$AP (Deluxe Version)"),
    ("Various Artists", "Cooking Songs"),
    ("Haim", "Forever - EP"),
    ("Lianne La Havas", "Is Your Love Big Enough?"),
    ("Michael Bublé", "To Be Loved"),
    ("Daughter", "If You Leave"),
    ("The xx", "xx"),
    ("Eminem", "Curtain Call"),
    ("Kendrick Lamar", "good kid, m.A.A.d city (Deluxe)"),
    ("Disclosure", "The Face - EP"),
    ("Palma Violets", "180"),
```

```
    ("Cody Simpson", "Paradise"),
    ("Ed Sheeran", "+ (Deluxe Version)"),
    ("Michael Bublé", "Crazy Love (Hollywood Edition)"),
    ("Bon Jovi", "Bon Jovi Greatest Hits - The Ultimate Collection"),
    ("Rita Ora", "Ora"),
    ("g33k", "Spabby"),
    ("Various Artists", "Annie Mac Presents 2012"),
    ("David Bowie", "The Platinum Collection"),
    ("Bridgit Mendler", "Ready or Not (Remixes) - EP"),
    ("Dido", "Girl Who Got Away"),
    ("Various Artists", "Now That's What I Call Disney"),
    ("The 1975", "Facedown - EP"),
    ("Kodaline", "The Kodaline - EP"),
    ("Various Artists", "100 Hits: Super 70s"),
    ("Fred V & Grafix", "Goggles - EP"),
    ("Biffy Clyro", "Only Revolutions (Deluxe Version)"),
    ("Train", "California 37"),
    ("Ben Howard", "Every Kingdom (Deluxe Edition)"),
    ("Various Artists", "Motown Anthems"),
    ("Courteeners", "ANNA"),
    ("Johnny Marr", "The Messenger"),
    ("Rodriguez", "Searching for Sugar Man"),
    ("Jessie Ware", "Devotion"),
    ("Bruno Mars", "Unorthodox Jukebox"),
    ("Various Artists", "Call the Midwife (Music From the TV Series)"
);
```

(The test data chosen happens to be the current 150 top iTunes albums at the time of writing!)

Now create the database using the following:

```
$ sqlite data/zftutorial.db < data/album-fixtures.sql
```

Some systems, including Ubuntu, use the command `sqlite3` ; check to see which one to use on your system.

> **Using PHP to create the database**
>
> If you do not have Sqlite installed on your system, you can use PHP to load the database using the same SQL schema file created earlier. Create the file `data/load_album_fixtures.php` with the following

contents:

```
 <?php
$db = new PDO('sqlite:' . realpath(__DIR__) . '/zftutorial.db');
$fh = fopen(__DIR__ . '/album-fixtures.sql', 'r');
while ($line = fread($fh, 4096)) {
    $db->exec($line);
}
fclose($fh);
```

Once created, execute it:

```
 $ php data/load_album_fixtures.php
```

This gives us a handy extra 150 rows to play with. If you now visit your album list at `/album`, you'll see a huge long list of 150+ albums; it's ugly.

Install zend-paginator

zend-paginator is not installed or configured by default, so we will need to do that. Run the following from the application root:

```
$ composer require zendframework/zend-paginator
```

Assuming you followed the Getting Started tutorial, you will be prompted by the zend-component-installer plugin to inject `Zend\Paginator`; be sure to select the option for either `config/application.config.php` or `config/modules.config.php`; since it is the only package you are installing, you can answer either "y" or "n" to the "Remember this option for other packages of the same type" prompt.

Manual configuration

If you are not using zend-component-installer, you will need to setup configuration manually. You can do this in one of two ways:

- Register the `Zend\Paginator` module in either `config/application.config.php` or `config/modules.config.php`. Make sure you put it towards the top of the module list, before any modules you have defined or third party modules you are using.

- Alternately, add a new file, `config/autoload/paginator.global.php`, with the following contents:

```
 <?php
use Zend\Paginator\ConfigProvider;

return [
    'service_manager' => (new
ConfigProvider())->getDependencyConfig(),
];
```

Once installed, our application is now aware of zend-paginator, and even has some default factories in place, which we will now make use of.

Modifying the AlbumTable

In order to let zend-paginator handle our database queries automatically for us, we will be using the DbSelect pagination adapter This will automatically manipulate and run a `Zend\Db\Sql\Select` object to include the correct `LIMIT` and `WHERE` clauses so that it returns only the configured amount of data for the given page. Let's modify the `fetchAll` method of the `AlbumTable` model, so that it can optionally return a paginator object:

```
namespace Album\Model;

use RuntimeException;
use Zend\Db\ResultSet\ResultSet;
use Zend\Db\Sql\Select;
use Zend\Db\TableGateway\TableGatewayInterface;
use Zend\Paginator\Adapter\DbSelect;
use Zend\Paginator\Paginator;

class AlbumTable
{


    public function fetchAll($paginated = false)
    {
        if ($paginated) {
            return $this->fetchPaginatedResults();
```

```
        }

        return $this->tableGateway->select();
    }

    private function fetchPaginatedResults()
    {

        $select = new Select($this->tableGateway->getTable());


        $resultSetPrototype = new ResultSet();
        $resultSetPrototype->setArrayObjectPrototype(new Album());


        $paginatorAdapter = new DbSelect(

            $select,

            $this->tableGateway->getAdapter(),

            $resultSetPrototype
        );

        $paginator = new Paginator($paginatorAdapter);
        return $paginator;
    }


}
```

This will return a fully configured `Paginator` instance. We've already told the `DbSelect` adapter to use our created `Select` object, to use the adapter that the `TableGateway` object uses, and also how to hydrate the result into a `Album` entity in the same fashion as the `TableGateway` does. This means that our executed and returned paginator results will return `Album` objects in exactly the same fashion as the non-paginated results.

Modifying the AlbumController

Next, we need to tell the album controller to provide the view with a `Pagination` object instead of a `ResultSet`. Both these objects can by iterated over to return hydrated `Album` objects, so we won't

need to make many changes to the view script:

```
public function indexAction()
{

    $paginator = $this->table->fetchAll(true);



    $page = (int) $this->params()->fromQuery('page', 1);
    $page = ($page < 1) ? 1 : $page;
    $paginator->setCurrentPageNumber($page);


    $paginator->setItemCountPerPage(10);

    return new ViewModel(['paginator' => $paginator]);
}
```

Here we are getting the configured `Paginator` object from the `AlbumTable`, and then telling it to use the page that is optionally passed in the querystring `page` parameter (after first validating it). We are also telling the paginator we want to display 10 albums per page.

Updating the View Script

Now, tell the view script to iterate over the `pagination` view variable, rather than the `albums` variable:

```
 <?php
$title = 'My albums';
$this->headTitle($title);
?>
<h1><?= $this->escapeHtml($title); ?></h1>
<p>
    <a href="<?= $this->url('album', ['action' => 'add']) ?>">Add new
album</a>
```

```
</p>

<table class="table">
    <tr>
        <th>Title</th>
        <th>Artist</th>
        <th> </th>
    </tr>
    <?php foreach ($this->paginator as $album) : <-- change here! ?>
        <tr>
            <td><?= $this->escapeHtml($album->title) ?></td>
            <td><?= $this->escapeHtml($album->artist) ?></td>
            <td>
                <a href="<?= $this->url('album', ['action' => 'edit',
'id' => $album->id]) ?>">Edit</a>
                <a href="<?= $this->url('album', ['action' => 'delete',
'id' => $album->id]) ?>">Delete</a>
            </td>
        </tr>
    <?php endforeach; ?>
</table>
```

Checking the `/album` route on your website should now give you a list of just 10 albums, but with no method to navigate through the pages. Let's correct that now.

Much like we created a custom breadcrumbs partial to render our breadcrumb in the navigation tutorial, we need to create a custom pagination control partial to render our pagination control just the way we want it. Again, because we are using Bootstrap, this will primarily involve outputting correctly formatted HTML. Let's create the partial in the `module/Application/view/partial/` folder, so that we can use the control in all our modules:

```
 <?php
<?php if ($this->pageCount): ?>
<div>
  <ul class="pagination">

  <?php if (isset($this->previous)): ?>
    <li>
      <a href="<?= $this->url($this->route, [], ['query' => ['page' =>
$this->previous]]) ?>">
        &lt;&lt;
```

```
      </a>
    </li>
  <?php else: ?>
    <li class="disabled">
      <a href="#">
        &lt;&lt;
      </a>
    </li>
  <?php endif ?>


  <?php foreach ($this->pagesInRange as $page): ?>
    <?php if ($page !== $this->current): ?>
      <li>
        <a href="<?= $this->url($this->route, [], ['query' => ['page'
=> $page]]) ?>">
          <?= $page ?>
        </a>
      </li>
    <?php else: ?>
      <li class="active">
        <a href="#"><?= $page ?></a>
      </li>
    <?php endif ?>
  <?php endforeach ?>


  <?php if (isset($this->next)): ?>
    <li>
      <a href="<?= $this->url($this->route, [], ['query' => ['page' =>
$this->next]]) ?>">
        &gt;&gt;
      </a>
    </li>
  <?php else: ?>
    <li class="disabled">
      <a href="#">
        &gt;&gt;
      </a>
    </li>
  <?php endif ?>
  </ul>
```

```
</div>
<?php endif ?>
```

This partial creates a pagination control with links to the correct pages (if there is more than one page in the pagination object). It will render a previous page link (and mark it disabled if you are at the first page), then render a list of intermediate pages (that are passed to the partial based on the rendering style; we'll pass that to the view helper in the next step). Finally, it will create a next page link (and disable it if you're at the end). Notice how we pass the page number via the `page` querystring parameter which we have already told our controller to use to display the current page.

To page through the albums, we need to invoke the paginationControl view helper to display our pagination control:

```
 <?php


?>
<?= $this->paginationControl(

    $this->paginator,

    'sliding',

    'partial/paginator',

    ['route' => 'album']
) ?>
```

The above echoes the `paginationControl` helper, and tells it to use our paginator instance, the sliding scrolling style, our paginator partial, and which route to use for generating links. Refreshing your application now should give you Bootstrap-styled pagination controls!
