Adding zend-navigation to the Album Module

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

In this tutorial we will use the zend-navigation component to add a navigation menu to the black bar at the top of the screen, and add breadcrumbs above the main site content.

Preparation

In a real world application, the album browser would be only a portion of a working website. Usually the user would land on a homepage first, and be able to view albums by using a standard navigation menu. So that we have a site that is more realistic than just the albums feature, lets make the standard skeleton welcome page our homepage, with the `/album` route still showing our album module. In order to make this change, we need to undo some work we did earlier. Currently, navigating to the root of your app (`/`) routes you to the `AlbumController` 's default action. Let's undo this route change so we have two discrete entry points to the app, a home page, and an albums area.

```
'home' => [
   'type' => Literal::class,
    'options' => [
        'route'    => '/',
        'defaults' => [
            'controller' => Controller\IndexController::class,
            'action'     => 'index',
        ],
    ],
],
```

(You can also now remove the import for the `Album\Controller\AlbumController` class.)

This change means that if you go to the home page of your application (`http://localhost:8080/` or

`http://zf-tutorial.localhost/`), you see the default skeleton application introduction. Your list of albums is still available at the `/album` route.

Setting Up zend-navigation

First, we need to install zend-navigation. From your root directory, execute the following:

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

Assuming you followed the Getting Started tutorial, you will be prompted by the zend-component-installer plugin to inject `Zend\Navigation`; 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\Navigation` 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/navigation.global.php`, with the following contents:
>
> ```
> <?php
> use Zend\Navigation\ConfigProvider;
>
> return [
>     'service_manager' => (new
> ConfigProvider())->getDependencyConfig(),
> ];
> ```

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

Configuring our Site Map

Next up, we need zend-navigation to understand the hierarchy of our site. To do this, we can add a `navigation` key to our configuration, with the site structure details. We'll do that in the `Application` module configuration:

```
return [

    'navigation' => [
        'default' => [
            [
                'label' => 'Home',
                'route' => 'home',
            ],
            [
                'label' => 'Album',
                'route' => 'album',
                'pages' => [
                    [
                        'label'  => 'Add',
                        'route'  => 'album',
                        'action' => 'add',
                    ],
                    [
                        'label'  => 'Edit',
                        'route'  => 'album',
                        'action' => 'edit',
                    ],
                    [
                        'label'  => 'Delete',
                        'route'  => 'album',
                        'action' => 'delete',
                    ],
                ],
            ],
        ],
    ],

];
```

This configuration maps out the pages we've defined in our Album module, with labels linking to the given route names and actions. You can define highly complex hierarchical sites here with pages and sub-pages linking to route names, controller/action pairs, or external uris. For more information, see the zend-navigation quick start.

Now that we have the navigation helper configured by our service manager and merged config, we can add the menu to the title bar to our layout by using the menu view helper:

```
 <?php
<div class="collapse navbar-collapse">
    <?php
    <?= $this->navigation('navigation')->menu() ?>
</div>
```

The navigation helper is provided by default with zend-view, and uses the service manager configuration we've already defined to configure itself automatically. Refreshing your application, you will see a working menu; with just a few tweaks however, we can make it look even better:

```
 <?php
<div class="collapse navbar-collapse">
    <?php
    <?= $this->navigation('navigation')
        ->menu()
        ->setMinDepth(0)
        ->setMaxDepth(0)
        ->setUlClass('nav navbar-nav') ?>
</div>
```

Here we tell the renderer to give the root `<ul>` the class of `nav` (so that Bootstrap styles the menu correctly), and only render the first level of any given page. If you view your application in your browser, you will now see a nicely styled menu appear in the title bar.

The great thing about zend-navigation is that it integrates with zend-router in order to highlight the currently viewed page. Because of this, it sets the active page to have a class of `active` in the menu; Bootstrap uses this to highlight your current page accordingly.

Adding Breadcrumbs

Adding breadcrumbs follows the same process. In our `layout.phtml` we want to add breadcrumbs

above the main content pane, so our users know exactly where they are in our website. Inside the container `<div>`, before we output the content from the view, let's add a breadcrumb by using the breadcrumbs view helper.

```
<?php
<div class="container">
    <?php
    <?= $this->navigation('navigation')->breadcrumbs()->setMinDepth(0)
?>
    <?= $this->content ?>
</div>
```

This adds a simple but functional breadcrumb to every page (we tell it to render from a depth of 0 so we see all page levels), but we can do better than that! Because Bootstrap has a styled breadcrumb as part of its base CSS, let's add a partial that outputs the `<ul>` using Bootstrap styles. We'll create it in the `view` directory of the `Application` module (this partial is application wide, rather than album specific):

```
<?php
<ul class="breadcrumb">
    <?php

    foreach ($this->pages as $key => $page):
    ?>
        <li>
            <?php

            if ($key < count($this->pages) - 1):
            ?>
                <a href="<?= $page->getHref() ?>"><?= $page->getLabel()
?></a>
            <?php

            else:
            ?>
                <?= $page->getLabel() ?>
            <?php endif; ?>
        </li>
    <?php endforeach; ?>
</ul>
```

Notice how the partial is passed a `Zend\View\Model\ViewModel` instance with the `pages`

property set to an array of pages to render.

Now we need to tell the breadcrumb helper to use the partial we have just written:

```
 <?php
<div class="container">
    <?php
    <?= $this->navigation('navigation')
            ->breadcrumbs()
            ->setMinDepth(0)
            ->setPartial('partial/breadcrumb') ?>
    <?= $this->content ?>
</div>
```

Refreshing the page now gives us a styled set of breadcrumbs on each page.
