

Vues

Pour terminer cette explication du lien entre les routes, les contrôleurs et les vues, nous allons développer la manière dont sont gérées les vues par Castopod.

Tailwind

Castopod utilise pour ses vues [le framework Tailwind](#) qui est un utilitaire CSS open-source qui va permettre de simplifier la gestion du style des pages. Vous pouvez retrouver le fichier de configuration à la racine de Castopod, dans le fichier *tailwind.config.js*. Dans ce fichier, on peut notamment avoir la liste des couleurs pour chaque thème, ou des classes par défaut pour certains éléments HTML.

Cette utilisation de Tailwind est directement lié au fonctionnement de Vite dans le mode développement. C'est pour cela que vous retrouverez au début de chaque vue ce morceau de code permettant entre autre de pouvoir utiliser Tailwind :

```
<?= service('vite')
->asset('styles/index.css', 'css') ?>

<?= service('vite')
->asset('js/app.ts', 'js') ?>
```

Une fonction *service* est utilisée ici, et pour mieux comprendre l'utilisation de cette fonction que vous retrouvez avec d'autres arguments dans les vues, nous vous invitons à vous rendre sur [la page où nous détaillons l'utilité de cette fonction](#).

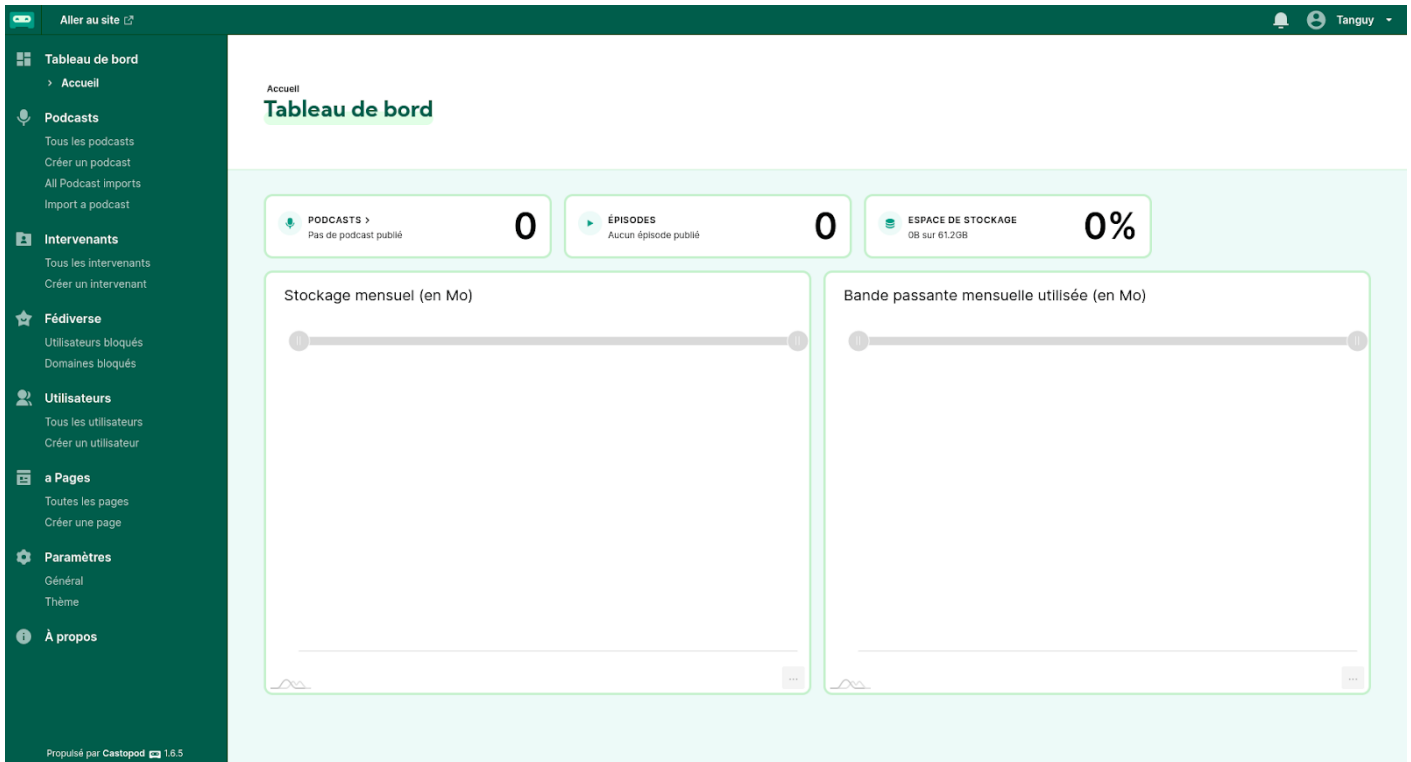
Layout

Les vues sont écrites en PHP, ce qui permet d'utiliser des fonctions de PHP et de CodeIgniter dans leur fichier. Cela permet par exemple d'utiliser un système de mise en page à l'aide de composant qu'on peut réutiliser dans plusieurs pages.

Prenons par exemple la page d'administration. Quel que soit la page sur laquelle vous êtes, vous retrouvez :

- Un bandeau supérieur, dans lequel vous avez le logo de Castopod, une redirection sur le site principal, etc.

- Un bandeau latéral à gauche, dans lequel vous retrouvez les différentes pages d'administration



Pour éviter de devoir réécrire dans chaque vue des pages d'administration, on va utiliser ce qui s'appelle des *layouts*, en écrivant des morceaux de page réutilisable. Pour les pages d'administration, les deux bandeaux sont `_partials/_nav_header.php` pour le bandeau supérieur et `_partials/_nav_aside.php` pour le bandeau latéral.

Il suffit ensuite d'indiquer sur les vues d'utiliser ces composants à l'aide de la fonction `include(nomLayout)`.

Pour ces deux composants, on aura donc dans la vue :

```
<?= $this->include('_partials/_nav_header') ?>
<?= $this->include('_partials/_nav_aside') ?>
```

Si vous allez voir la vue **dashboard**, vous ne verrez pas tout de suite ces deux lignes, elles sont en fait incluses dans un composant plus général, nommé `_layout`, qui va gérer d'autres éléments que l'on retrouve sur toutes les pages d'administrations (comme le titre de la page par exemple).

Si vous souhaitez ajouter des composants à votre vue en plus des *layouts* inclus, vous devrez les placer dans une section à part à l'aide de la fonction

```
<?= $this->section('nonSection') ?>*
```

Par exemple, pour la vue de création d'épisode depuis la page d'administration, voici ce que nous avons :

```

<?= $this->extend('_layout') ?>

<?= $this->section('title') ?>
<?= lang('Episode.create') ?>
<?= $this->endSection() ?>

<?= $this->section('pageTitle') ?>
<?= lang('Episode.create') ?>
<?= $this->endSection() ?>

<?= $this->section('content') ?>
    // Formulaire de création d'épisode
<?= $this->endSection() ?>

```

On peut voir que la page indique le titre de la page dans la section *title*, mais si vous vous souvenez bien de *_layout*, nous avons indiqué que le titre de la page était géré dans ce composant.

En vérité, il va générer le titre de la page à partir d'un texte fixe (qui est 'Castopod Admin' par défaut), ainsi que de la section de la page qui l'appelle. Cela se fait avec la fonction *renderSection*. Voici ce qu'on retrouve dans le fichier *_layout* :

```

<title><?= $this->renderSection('title') ?> | Castopod Admin</title>

```

Variable

Nous avons précédemment qu'il était possible de [charger une vue avec des données à l'aide du contrôleur](#), voyons maintenant comment utiliser ces données dans la page.

Si on reprend la page d'accueil, dans le contrôleur nous avons cette fonction pour charger la page :

```

namespace App\Controllers;

use App\Models\PodcastModel;
use CodeIgniter\HTTP\RedirectResponse;

class HomeController extends BaseController
{

```

```

public function index(): RedirectResponse | string
{
    ...

    // Création d'un objet data, qui va contenir les podcasts récupérés
    $data = [
        'podcasts' => $allPodcasts,
        'sortBy'   => $sortBy,
    ];

    // Charger la vue avec l'objet data
    return view('home', $data);
}
}

```

On voit ici qu'une variable *\$data* est créé avec deux attributs : *podcasts* et *sortBy*. Le nom de ces attributs est important car c'est ce nom qui pourra être utilisé pour référer directement à leur valeur dans la vue.

Ainsi, pour afficher les différents podcasts récupérés par le contrôleur, il suffit de faire une boucle sur tous les podcasts (en vérifiant au passage que cette variable n'est pas vide, ce qui signifierai qu'aucun podcast n'a encore été créé. Voici ce qu'on trouve dans la vue correspondant à la page d'accueil de Castopod :

```

<div class="grid gap-4 mt-4 grid-cols-cards">
    <?php if ($podcasts): ?>
        <?php foreach ($podcasts as $podcast): ?>
            // Gestion de chaque podcast trouvé
        <?php endforeach; ?>
    <?php else: ?>
        // Gestion si aucun podcast n'a été trouvé
        <p class="italic"><?= lang('Home.no_podcast') ?></p>
    <?php endif; ?>
</div>

```

Pour rappel, *\$podcasts* contenait la liste des podcasts créés, sous forme de l'entité *Podcast*. Donc chaque élément de cette liste contenait les informations du podcast, c'est-à-dire le nom, l'identifiant, le lien, etc.

On peut directement accéder à ces informations en indiquant l'attribut à accéder dans la variable. Pour accéder au nom du premier podcast par exemple, on pourra y accéder comme cela :

```
$podcasts[0]->title
```

Revision #6

Created 10 February 2024 17:20:09 by tanguy

Updated 11 February 2024 15:00:56 by tanguy