

# Routes

Comme expliqué auparavant, Castopod s'appuie sur un modèle MVC qui permet d'accéder à des pages du site grâce à des contrôleurs qui vont charger des vues. Si vous n'avez pas compris un seul mot, nous vous renvoyons vers l'article qui correspond à [l'explication de ce principe](#).

Le projet est divisé en deux dossiers principaux, le dossier **App** et le dossier **Modules**, le premier étant plutôt destiné aux pages qui vont être accessibles à n'importe qui, tandis que le deuxième est destiné à une utilisation particulière comme de l'administration ou de l'authentification. Là aussi, si vous n'avez pas tout compris, nous vous redirigeons vers l'article correspondant à [l'organisation du projet](#).

## Définition d'une route

Pour développer sur la manière dont les routes sont agencées dans le projet, nous allons nous intéresser dans un premier à la structure d'une route. Prenons par exemple la route vers la page d'accueil (accessible à l'emplacement `app/Config/Routes.php`) :

```
$routes->get('/', 'HomeController', [  
    'as' => 'home',  
]);
```

Cette configuration indique quatre éléments :

- La route est accessible via le protocole GET (et non POST ou un autre)
- Elle concerne l'index '/', c'est-à-dire la racine du projet, lorsque l'on tape juste l'URL de l'hébergement (par défaut `localhost:8080`)
- Elle fait appelle au contrôleur **HomeController**, avec la fonction particulière (il n'y a pas de fonction d'indiquée)
- Le nom de cette route est **home**. On pourra donc utiliser ce nom si on souhaite configurer une redirection dans une autre route ou un contrôleur.

La gestion de l'affichage est ensuite gérée par le contrôleur associé à la route, ici **HomeController**, qui en fonction de la fonction appelée par la route pourra aller chercher la vue recherchée.

Si maintenant on souhaite avoir une route sur l'URL avec comme chemin `/health`, qui permettra de savoir si Castopod rencontre un problème ou non.

La structure sera assez similaire, il faudra juste changer l'URL correspondant à la route, et indiquer

une fonction particulière du contrôleur :

```
$routes->get('/health', 'HomeController::health', [  
    'as' => 'health',  
]);
```

Avec cette route, lorsque l'utilisateur accèdera à la page *URL/health*, cela chargera la fonction *health()* du contrôleur **HomeController**.

Enfin, il est possible d'indiquer un paramètre à donner au contrôleur lors de la connexion, en utilisant le caractère \$.

On peut appliquer des filtres à ce paramètre en indiquant son type, et en définissant ce type au début du fichier *route*.

Imaginons que la fonction *health(int ID)* du contrôleur nécessite un paramètre numérique strictement inférieur à 1000. Nous pourrions donc commencer par créer un filtre **nombre**, et appliquer ce filtre au paramètre indiqué par la route :

```
$routes->addPlaceholder('nombre', '[0-9]{1,3}');
```

Nous avons ici créer un filtre **nombre**, qui indique que chaque caractère doit être compris entre 0 et 9, et la taille de la chaîne de caractère ne peut pas être plus grand que 3 (donc numériquement qui soit inférieure ou égale 999).

Pour créer la route, il suffira donc d'indiquer le paramètre lors de la déclaration :

```
$routes->get('health/(:nombre)', 'HomeController::health/$1');
```

## Définition d'un groupe

Avec un ensemble de routes, il est possible de les regrouper sous un groupe. Cela peut être pratique si ces routes possèdent un attribut commun, comme le fait qu'elles nécessitent un même paramètre ou qu'elles font parties d'un endroit particulier sur le site (comme la partie administration par exemple).

Prenons comme exemple l'accès à une page de podcast pour un auditeur lambda. Pour accéder à la page de ce podcast, Castopod utilise le nom du podcast qui est une chaîne de caractère de taille maximale de 32. Pour cela, lorsque l'utilisateur clique sur un podcast, on accède à la page du podcast sous la forme *URL/@nom\_du\_podcast*.

Dans les routes, cela est représenté par un groupe de route, qui récupère ce premier paramètre de l'URL (*@nom\_du\_podcast*) via un filtre, et qui développe ensuite quel contrôleur il doit appelé en fonction de la page de ce podcast.

Pour être plus concret, voici ce que fait concrètement Castopod. Tout d'abord, il crée un filtre **podcastHandle** pour le nom du podcast :

```
$routes->addPlaceholder('podcastHandle', '[a-zA-Z0-9\_]{1,32}');
```

Une fois ce filtre créé, un groupe est indiqué pour regrouper toutes les routes possibles lorsqu'un podcast a été sélectionné par l'utilisateur :

```
$routes->group('@(:podcastHandle)', static function ($routes): void {  
    ...  
})
```

On a regroupé ici tous les liens qui commencent par un *@nom\_du\_podcast*, donc dès que l'utilisateur voudra accéder à une page qui commence par *URL/@nom\_du\_podcast/...*, Castopod ira récupérer les routes dans ce groupe.

Imaginons que l'utilisateur souhaite accéder à la page d'un podcast, l'URL sera juste *URL/@nom\_du\_podcast*. Cela signifie donc que l'utilisateur accède à la racine de ce groupe, et comme précédemment, pour indiquer la racine, on utilise l'index '/'.

Cela donnera donc sous forme de routes :

```
$routes->group('@(:podcastHandle)', static function ($routes): void {  
    $routes->get('/', 'PodcastController::activity/$1', [  
        'as' => 'podcast-activity',  
    ]);  
})
```

Ici, Castopod utilise le contrôleur **PodcastController** pour gérer l'affichage de la page, via la fonction *activity()*. Et si on souhaite accéder à une autre page du podcast, comme par exemple la page liée aux informations supplémentaires (la page à propos) via le lien *URL/@nom\_du\_podcast/about*, comme précédemment, il suffira de rajouter une route dans le groupe :

```
$routes->group('@(:podcastHandle)', static function ($routes): void {  
    $routes->get('/', 'PodcastController::activity/$1', [  
        'as' => 'podcast-activity',  
    ]);  
  
    $routes->get('about', 'PodcastController::about/$1', [  
        'as' => 'podcast-about',  
    ]);  
})
```

Enfin, il est possible dans les groupes d'indiquer également l'espace de travail (c'est-à-dire l'emplacement) des contrôleurs à utiliser dans les routes. Par exemple, pour indiquer que les routes liées à l'authentification (qui dans l'exemple commencent par *URL/authentification/...*) utiliseront les contrôleurs du dossier **modules/Auth/Controllers**, on peut ajouter ce paramètre :

```
$routes->group(
    'authentification',
    [
        'namespace' => 'Modules\Auth\Controllers',
    ],
    static function ($routes): void {
        ...
    }
);
```

Pour aller plus en profondeur dans la gestion des routes sur CodeIgniter, nous vous invitons à lire [la documentation liée au routing](#) qui développe sur les autres paramètres possibles pour une route. La gestion des différents paramètres est ensuite gérée par les contrôleurs, ainsi que la gestion de l'affichage de la page. Nous détaillons ce fonctionnement sur [cette page](#).

---

Revision #11

Created 10 February 2024 15:44:40 by tanguy

Updated 10 February 2024 20:30:37 by tanguy