UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Framework PHP : LARAVEL

Les bases du routage

Mouhamed DIOP - Ingénieur de Conception en Informatique (ESP)

Objectifs spécifiques

A l'issue de cette séquence, l'étudiant devrait :

- Savoir ce que c'est que le routage
- Connaitre les principaux fichiers de routage et leurs rôles
- Pouvoir afficher la liste des routes
- Pouvoir définir des routes web simples
- Pouvoir définir des routes web paramétrées
- Pouvoir définir des routes web de redirection
- Pouvoir associer des routes web à des vues
- Pouvoir associer des routes web à des méthodes de contrôleurs

Qu’est-ce que le routage ?

Une route définit un point d’entrée dans une application

- Le routage est le mécanisme par lequel on rend accessible aux utilisateurs les fonctionnalités de l’application développée
- Il permet d’associer des URL à des fonctionnalités applicatives
- Il définit ainsi les URL auxquelles une application va répondre

Le routage permet donc de définir l’opération à effectuer par l’application à chaque fois qu’un visiteur essaye d’accéder à une ressource (URL)

Qu’est-ce que le routage ?

Depuis la version 5.2 de Laravel, les fichiers de routage sont stockés dans le répertoire *routes/* et sont classés par catégorie :

- **Les routes web** : utilisées pour la transmission de pages aux navigateurs Web (ordinateurs de bureau et mobiles), via le protocole HTTP ou HTTPS. Elles sont définies dans le fichier web.php
- **Les routes d’API** : principalement utilisées pour les API de type JSON ou REST, accessibles par les navigateurs ou d’autres applications. Elles sont définies dans le fichier api.php
- **Les routes « channels » :** utilisées pour définir des canaux de notification. Elles sont définies dans le fichier channels.php
- **Les routes pour console :** utilisées pour créer des commandes **Artisan** basées sur des **closures** (pour des tâches d'administration rapides). Elles sont définies dans le fichier console.php

Listing des routes

Pour voir la liste de toutes les routes disponibles sur une application

- Il suffit d'ouvrir la console depuis la racine du projet et de taper la commande suivante :

  php artisan route:list

- Cette commande produira un tableau qui montre tous les points d'entrée, ainsi que les types de requêtes HTTP acceptés
- Dans ce cours, nous nous intéresserons uniquement aux routes web, destinées aux visiteurs d'un site. Elles doivent toutes être définies dans le fichier ***routes/web.php***

Création d'une route

Il suffit de renseigner :

- Par quelle méthode HTTP on accède à la page (GET, POST, …)
- L'URL qu'on souhaite attribuer à la page
- La fonction à exécuter (closure) quand la page est demandée

Exemple : Afficher « Hello World » à l'utilisateur quand il accède à l'adresse http://localhost/home

```
Route::get('/home', function () {
    return 'Hello World';
});
```

Création d'une route : autre exemple

Exemple : Afficher un message informant l'utilisateur du succès de l'ajout d'un utilisateur quand il envoie une requête POST (soumission d'un formulaire) à l'adresse http://localhost/user/create

```
Route::post('/user/create', function () {
    $reponse = '<h1>Ajout Utilisateur</h1>' ;
    $reponse .= '<p>Utilisateur ajouté avec succès</p>' ;
    return $reponse;
});
```

Création de routes et méthodes HTTP

La classe Route peut gérer tous les types de requêtes HTTP

- Route::get($uri, $callback) : gère les requêtes de type GET
- Route::post($uri, $callback) : gère les requêtes de type POST
- Route::put($uri, $callback) : gère les requêtes de type PUT
- Route::patch($uri, $callback) : gère les requêtes de type PATCH
- Route::delete($uri, $callback) : gère les requêtes de type DELETE
- Route::options($uri, $callback) : gère les requêtes de type OPTIONS

Création d'une route : any et match

Il est possible d'associer une URL à plusieurs types de requêtes HTTP

- Utilisation de la méthode ***match***

L'accès à http://localhost/home affiche « Hello World », que le type d'accès soit par GET ou POST

```
Route::match([get, post], '/home', function () {
    return 'Hello World';
});
```

- Utilisation de la méthode ***any***

L'accès à http://localhost/home affiche « Hello World », peu importe le type d'accès

```
Route::any('/home', function () {
    return 'Hello World';
});
```

Création d'une route paramétrée

Il est possible d'ajouter un paramètre dans la définition de la route

- Il suffit d'encadrer le paramètre par des accolades
- Un contrôle peut être effectué sur le type du paramètre attendu
- Plusieurs paramètres peuvent être définis dans une route

Exemple : Afficher « Bonjour fatou » à l'utilisateur quand il accède à l'adresse http://localhost/saluer/fatou

```
Route::get('/saluer/{nom}', function ($nom) {
    $reponse = 'Bonjour ' . $nom ;
    return $reponse;
});
```

Création d'une route paramétrée

Il est possible d'ajouter une contrainte sur le format du paramètre

- Il faudra utiliser la méthode where de la classe Route
- La contrainte se définit à l'aide d'expressions régulières (regex)
- La méthode where peut prendre plusieurs arguments au cas où la route contiendrait plusieurs paramètres à contraindre

Exemple : Afficher « Utilisateur N°1 » à l'utilisateur quand il accède à l'adresse http://localhost/users/1

```
Route::get('/users/{id}', function ($id) {
    $reponse = 'Utilisateur N°' . $id;
    return $reponse;
})->where('id', '[0-9]+') ;
```

Cette route ne sera pas activé si on donne autre chose qu'un entier pour l'ID

Création d'une route paramétrée

Un paramètre optionnel peut être ajouté à une route

- Il peut arriver qu'on ait besoin de spécifier un paramètre dans une route, mais que sa présence soit optionnel
- On peut satisfaire ce besoin en ajoutant un point d'interrogation après le nom du paramètre
- Il faudra tout de même donner une valeur par défaut à la variable correspondante dans la closure

Exemple : Afficher « Bonjour fatou » à l'utilisateur quand il accède à l'adresse http://localhost/saluer/fatou. L'accès à http://localhost/saluer affichera « Bonjour Inconnu »

```
Route::get('/saluer/{nom?}', function ($nom = 'Inconnu') {
    $reponse = 'Bonjour ' . $nom ;
    return $reponse;
});
```

Création d'une route de redirection

Il est possible de rediriger l'utilisateur vers une autre page

- Exemple : Rediriger l'utilisateur vers http://localhost/accueil quand il essaye d'accéder à http://localhost/home

```
Route::redirect('/home', '/accueil');
```

Il est possible de préciser le code d'état HTTP durant la redirection

- Par défaut, le code d'état vaut 302
- Pour changer cette valeur en 301 par exemple, il faut faire :

```
Route::redirect('/home', '/accueil', 301);
```

Il est possible de faire une redirection permanente en faisant :

```
Route::permanentRedirect('/home', '/accueil');
```

Création d'une route pour une vue

Il est possible de faire correspondre une URL à une vue

- Il n'est pas aisé d'écrire du code HTML directement dans les closures
- Laravel propose un mécanisme permettant de faire appel à des fichiers spéciaux (les vues) depuis les closures
- Ces vues ne sont rien d'autres que des fichiers HTML avec des portions de codes dynamiques. Nous les aborderons plus tard.

Exemple : Afficher la vue « welcome » à l'utilisateur quand il accède à l'adresse http://localhost/

```
Route::get('/', function () {
    return view('welcome');
});
```

Notez que l'exemple correspond au contenu par défaut du fichier de routage **routes/web.php**

Création d'une route pour une vue

Il est possible de faire correspondre une URL à une vue

- Si la route a seulement besoin de retourner la vue sans effectuer un quelconque traitement, on peut ne pas utiliser de closures
- Il suffit d'invoquer la méthode view de la classe Route.

Exemple : Afficher la vue « welcome » à l'utilisateur quand il accède à l'adresse http://localhost/

```
Route::view('/', 'welcome');
```

Création d'une route pour un contrôleur

Il est possible de faire correspondre une URL à une méthode d'un contrôleur

- La notion de contrôleur sera abordée plus tard
- A ce niveau, notez juste la syntaxe permettant d'associer une URL à une action particulière d'un contrôleur

Exemple : Demander à Laravel d'invoquer la méthode getList du contrôleur UserController quand l'utilisateur accède à l'adresse http://localhost/users

```
Route::get('/users', 'UserController@getList');
```

Liens utiles

Pour aller plus loin…

- https://laravel.com/docs/6.x/routing
- https://laracasts.com/series/laravel-from-scratch-2018/episodes/3
- https://asklaravel.com/88/routing/what-are-laravel-routes/
- https://openclassrooms.com/fr/courses/3613341-decouvrez-le-framework-php-laravel/3616576-le-routage-et-les-facades
- https://fr.wikipedia.org/wiki/Expression_r%C3%A9guli%C3%A8re
- https://www.lucaswillems.com/fr/articles/25/tutoriel-pour-maitriser-les-expressions-regulieres
- https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/916990-les-expressions-regulieres-partie-1-2