

## Système de routage sous laravel (suite)

### Remarque :

pour afficher la liste des routes , on utilise la commande artisan suivante :

```
php artisan route:list
```

### La méthode Route ::view

Si votre itinéraire ne doit renvoyer qu'une vue , vous pouvez utiliser la méthode **Route::view**. La méthode view accepte un URI comme premier argument et un nom de vue comme second argument.

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

De plus, vous pouvez fournir **un tableau de données** à transmettre à la vue en tant que troisième argument facultatif :

```
Route::view('/index', 'welcome',[ 'id' => 100]) ;
```

### Exercice :

Utiliser la route précédente et récupérer l'id dans la vue 'welcome.blade.php' pour l'afficher comme valeur d'une zone de texte en lecture seule.

### La méthode Route::redirect

Si vous définissez une route qui redirige vers un autre URI, vous pouvez utiliser la méthode **Route::redirect** :

```
Route::redirect('/test2', '/test1') ;
```

Par défaut, Route::redirect renvoie un **code d'état 302**. Vous pouvez personnaliser le code d'état à l'aide du troisième paramètre facultatif :

```
Route::redirect('/test2', '/test1',301) ;
```

Ou, vous pouvez utiliser la méthode **Route::permanentRedirect** pour renvoyer un code d'état 301:

```
Route::permanentRedirect('/test2', '/test1') ;
```

### La méthode Route ::fallback

En utilisant la méthode Route::fallback, vous pouvez définir une route qui sera exécutée lorsqu'aucune autre route ne correspond à la demande entrante. En règle générale, les requêtes non gérées afficheront automatiquement une page "404" via le gestionnaire d'exceptions de votre application. Cependant, comme vous définissez généralement la fallbackroute dans votre fichier routes/web.php, tous les middleware du groupe middleware web s'appliqueront à la route. Vous êtes libre d'ajouter un middleware supplémentaire à cette route si nécessaire :

```
Route::fallback(function() { .. } ) ;
```

### Exemple :

```
Route::fallback(function(){
```

```
echo "Page de la route ".Route::currentRouteName()." n'existe pas!";
});
```

## Les Contrôleurs :

Les contrôleurs se trouvent dans le dossier « App/Http/Controllers ». Un contrôleur est représenté par une classe PHP contenant plusieurs méthodes. Pour créer un nouveau contrôleur on tape la commande suivante :

```
php artisan make:controller NomControleur
```

Exercice :

1- Créer un contrôleur 'CalculController'. Puis ajouter deux méthodes :

- une méthode somme : qui a deux nombres en paramètres et qui calcule la somme et retourne : 'la somme est ....'

- une méthode produit : qui a deux nombres en paramètres et qui calcule le produit et retourne : 'le produit est ...'

2- Dans le fichier 'web.php' , ajouter deux routes , chacune vers une méthode du contrôleur .Attention, les routes doivent avoir deux paramètres.

3- tester le bon fonctionnement.

## Les vues

Bien sûr, il n'est pas pratique de renvoyer des chaînes de documents HTML entières directement à partir de vos routes et de vos contrôleurs. Heureusement, les vues offrent un moyen pratique de placer tout notre code HTML dans des fichiers séparés. Les vues séparent la logique de votre contrôleur/application de votre logique de présentation et sont stockées dans le répertoire ressources/views. Une vue simple pourrait ressembler à ceci :

```
<!-- Vue stockée dans ressources/views/v1.blade.php -->
<html>
<body>
<h1>Bonjour, {{ $name }}</h1>
</body>
</html>
```

Vous pouvez créer une vue en plaçant un fichier avec l'extension .blade.php dans le répertoire ressources/views de votre application. L'extension .blade.php informe le framework que le fichier contient un modèle Blade .

Une fois que vous avez créé une vue, vous pouvez la renvoyer à partir de l'une des **routes** ou des **contrôleurs** de votre application à l'aide de la méthode view:

```
return view('v1', ['name' => 'Zineb']);
```

Exercice :

1- Modifier la méthode somme du contrôleur ( de l'exercice précédent) pour qu'elle retournera la somme directement au lieu de toute une phrase contenant la somme.

- 2- Créer une vue `affichage.blade.php` qui contient une `div` où on affichera la somme.
- 3- tester le bon fonctionnement

**Remarque :** Répertoires de vue imbriqués

Les vues peuvent également être imbriquées dans des sous-répertoires du répertoire `resources/views`. L'opérateur `"."` peut être utilisée pour référencer des vues imbriquées.

Exemple : si votre vue est stockée dans `resources/views/admin/v1.blade.php`, vous pouvez la renvoyer depuis l'une des routes/contrôleurs de votre application comme suit :

```
return view('admin.profile', ...);
```

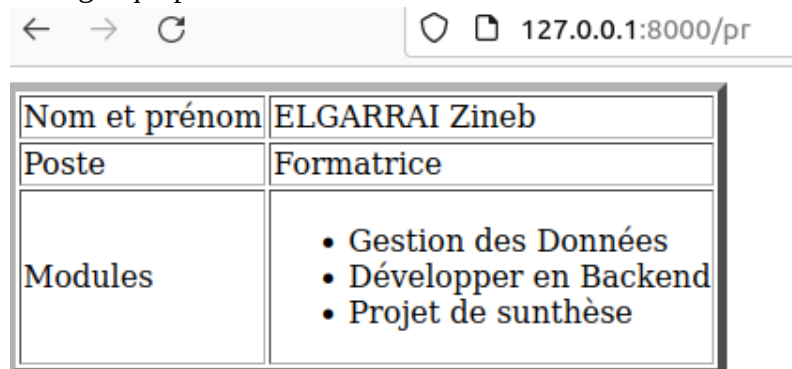
## TP 1 :

- 1- Créer un contrôleur 'ControllerTP1'. Définir la méthode étudiant suivante :

```
public function etudiant(){  
    $data=[];  
    $data['nom']='ELGARRAI';  
    $data['prenom']='Zineb';  
    $data['poste']='Formatrice';  
    $data['Modules']=['Gestion des Données','Développer en Backend','Projet de sunthèse'];  
    return view('affichage',$data);  
}
```

- 2- Définir une route correspondante

- 3- créer une vue 'affichage' qui permet d'afficher les informations comme suit :



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/pr`. The main content area displays a table with the following data:

Nom et prénom	ELGARRAI Zineb
Poste	Formatrice
Modules	<ul style="list-style-type: none"><li>• Gestion des Données</li><li>• Développer en Backend</li><li>• Projet de sunthèse</li></ul>