Manipulation des vues:

A. Création

1.Introduction :

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 **resources/views**. Une vue simple pourrait ressembler à ceci :

```
<!—vue stockée dans resources/views/home.blade.php-->
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1> Home page , {{$name}} </h1>

</body>
</html>
```

Nous pouvons la renvoyer en utilisant le helper global **view** comme ceci :

```
Route::get('/', function () {
return view('home', ['name' => 'James']);
});
```

L'extension **.blade.php** informe le **framework** que le fichier contient un modèle **Blade**. Les modèles de Blade contiennent du HTML ainsi que des directives Blade qui vous permettent de faire facilement écho des valeurs, de créer des instructions "if", d'itérer sur les données, etc.

2.Répertoires de vue imbriqués :

Les vues peuvent également être imbriquées dans des sous-répertoires du répertoire **resources/views**. La notation "Point" peut être utilisée pour référencer des vues imbriquées. Par exemple, si votre vue est stockée dans **resources/views/admin/profile.blade.php**, vous pouvez la renvoyer depuis l'une des routes/contrôleurs de votre application comme suit :

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

3.Déterminer si une vue existe

Si vous avez besoin de déterminer si une vue existe, vous pouvez utiliser la façade **View**. La méthode **exists** retournera **true** si la vue existe :

```
use Illuminate\Support\Facades\View;
if (View::exists('emails.customer')) {
//
 }
```

B. Transmission des données

1.Introduction :

Comme vous l'avez vu dans les exemples précédents, vous pouvez passer un tableau de données aux vues pour rendre ces données disponibles à la vue :

```
 return view('home', ['name' => 'Victoria']);
```

Lors de la transmission d'informations de cette manière, les données doivent être un tableau avec des paires **clé/valeur**. Après avoir fourni des données à une vue, vous pouvez ensuite accéder à chaque valeur de votre vue à l'aide des clés de données, telles que **{{$name}}**

Au lieu de transmettre un tableau complet de données à la fonction helper **view**, vous pouvez utiliser la méthode **with** pour ajouter des éléments de

données individuels à la vue. La méthode **with** renvoie une instance de l'objet **view** afin que vous puissiez continuer à enchaîner les méthodes avant de renvoyer la vue :

```
return view('home')
       ->with('name','Victoria')
       ->with('occupation', 'Astronaut');
```

2.Partage de données avec toutes les vues

Parfois, vous devrez peut-être partager des données avec toutes les vues rendues par votre application. Vous pouvez le faire en utilisant la méthode **share** de la façade **View**. En règle générale, vous devez placer des appels à la méthode **share** dans la méthode d'un fournisseur de services boot. Vous êtes libre de les ajouter à la classe **App\Providers\AppServiceProvider** ou de générer un fournisseur de services distinct pour les héberger :

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\View;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Response;

class AppServiceProvider extends ServiceProvider
{
    public function register()
    {
        //
    }

    public function boot()
    {

            View::share('key', 'value');

    }
}
```

C. Composeurs des vues

1. Introduction

Les composeurs de vue sont des rappels ou des méthodes de classe qui sont appelées lorsqu'une vue est rendue. Si vous avez des données que vous souhaitez lier à une vue chaque fois que cette vue est rendue, un composeur de vue peut vous aider à organiser cette logique en un seul emplacement. Les composeurs de vues peuvent s'avérer particulièrement utiles si la même vue est renvoyée par plusieurs routes ou contrôleurs au sein de votre application et a toujours besoin d'un élément de données particulier.

• En règle générale, les composeurs de vues seront enregistrés auprès de l'un des fournisseurs de services de votre application.

Nous utiliserons la méthode **composer** de la façade **View** pour enregistrer le composeur de vue :

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\View;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Response;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
```

```
        View::composer('pages.home', function ($view) {
            $view->with('title', 'how to be a great programmer');
        });

    }
}
```

Maintenant que nous avons enregistré le composeur, le contenu de la fonction de rappel sera exécutée à chaque **'home'** de la vue est rendu.

Vous pouvez attacher un composeur de vues à plusieurs vues à la fois en passant un tableau de vues comme premier argument à la méthode composer :

```
public function boot()
    {
        View::composer(['pages.home','pages.welcome'], function ($view) {
            $view->with('mytitle','how to be a great programmer');
        });

    }
```

La méthode **composer** accepte également le caractère * comme caractère générique, vous permettant d'attacher un composeur à toutes les vues :

```
 View::composer('*', function ($view) {
//
 });
```