

```
string sInput;  
int iLength, iN;  
double dblTemp;  
bool again = true;
```

```
while (again) {  
    iN = -1;  
    again = false;  
    getline(cin, sInput);  
    system("cls");  
    stringstream(sInput) >> dblTemp;  
    iLength = sInput.length();  
    if (iLength < 4) {  
        again = true;  
        continue;  
    } else if (sInput[iLength - 3] != '.') {  
        again = true;  
        continue;  
    } while (++iN < iLength) {  
        if (isdigit(sInput[iN])) {  
            continue;  
        } else if (iN == (iLength - 3)) {  
            continue;  
        }  
    }  
}
```

# LARAVEL

---

## Module 04 Blade



# Au programme dans ce module

## ❑ Chapitre 1 : Qu'est ce que Laravel Blade ?

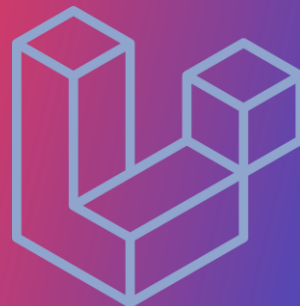
- Une surcouche PHP Frontend
- Des raccourcis pratiques
- Place à la pratique

## ❑ Chapitre 2 : Blade - Layout & include

- La gestion pyramidale
- Place à la pratique

## ❑ Chapitre 3 : Les assets frontend

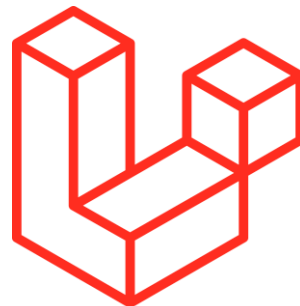
- Laravel Mix – Présentation
- Laravel Mix – Installation
- Laravel Mix
- Sass / CSS
- JS
- Images et fichiers publics
- Pratiquer c'est mieux



# Qu'est ce que Laravel Blade ?

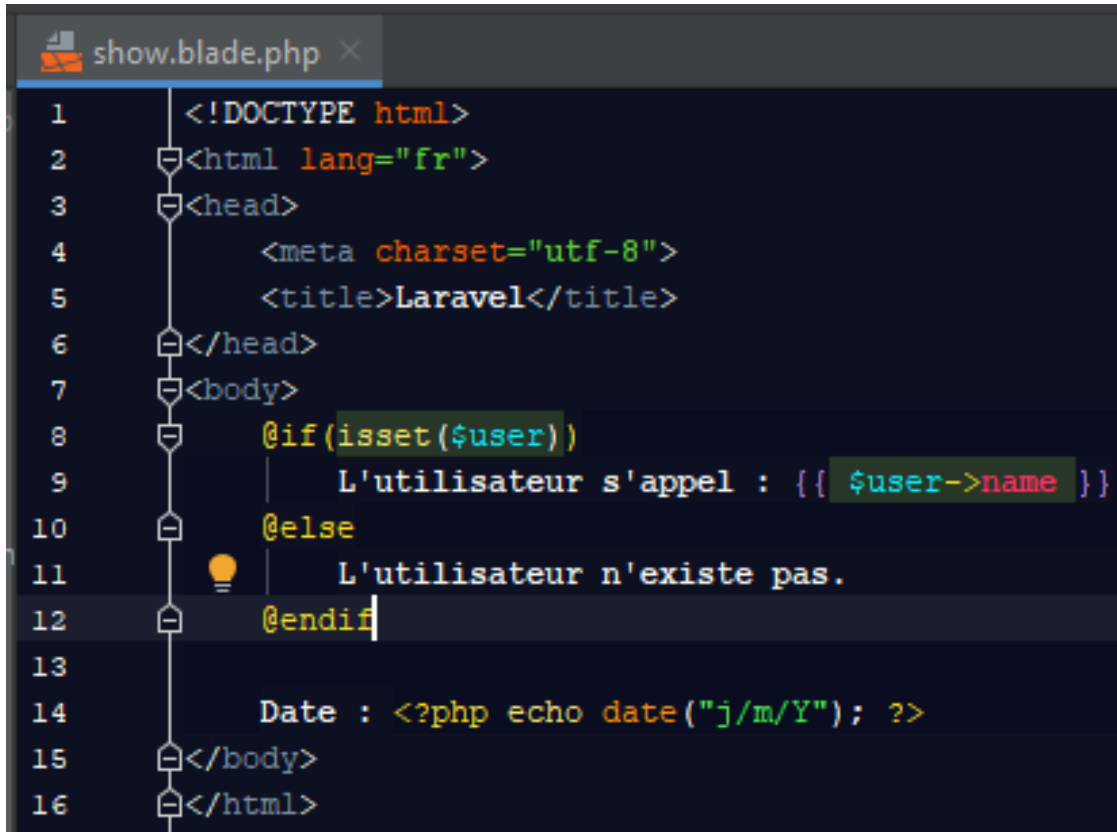


## Chapitre 1



Qu'est ce que Laravel Blade ?

# Une surcouche PHP Frontend



```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8">
5     <title>Laravel</title>
6 </head>
7 <body>
8     @if(isset($user))
9         L'utilisateur s'appel : {{ $user->name }}
10    @else
11        L'utilisateur n'existe pas.
12    @endif
13
14    Date : <?php echo date("j/m/Y"); ?>
15 </body>
16 </html>
```

- **Simple et rapide** à utiliser
- **Limite les répétitions** de code PHP et HTML
- Une extension unique : « **.blade.php** »
- Un **cache optimisé et géré par Laravel** visible dans « /storage/framework/views »
- De nombreux raccourcis Blade distinguable par des @
- Création de **ses propres directives Blade** !
  - Ajoutez-les dans AppServiceProvider.php
  - [Voir la doc](#)
- Bien d'autres avantages à découvrir sur la [doc](#)



Qu'est ce que Laravel Blade ?

## Des raccourcis pratiques

→ [Voir la doc](#)

```
@if(str_contains('@', $log_id))
|   L'identifiant "{{ $log_id }}" est un email.
@elseif($log_id === NULL)
|   Il n'y pas d'identifiant.
@else
|   L'identifiant est un surnom.
@endif
```

```
<ul>
|   @foreach($users as $user)
|       <li>{{ $user->name }}</li>
|   @endforeach
</ul>
```

```
<ul>
|   @for($page=0; $page<=5; $page++)
|       <li>
|           <a href="@route('features.show', $page)">
|               Vers la page {{ $page }}
|           </a>
|       </li>
|   @endfor
</ul>
```

```
{{ $var }} <=> <?php echo e($var); ?>
{!! $var !!} <=> <?php echo $var; ?>
{{-- Commentez moi --}}
```

```
@route('ROUTE_NAME', $var)
@route('ROUTE_NAME', [$var1, $var2])
```

```
@auth
|   Authenticated
@endauth

@auth('admin')
|   Authenticated with the admin guard
@endauth

@guest
|   Not authenticated
@endguest
```



## Place à la pratique

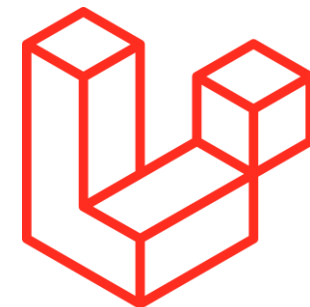
- ❖ Créez une vue Blade « users.index » (en n'oubliant pas la route, le contrôleur et la méthode pour y accéder).
- ❖ Transmettre, depuis le contrôleur, une variable (tableau PHP) contenant 2 strings et 1 nombre à votre vue Blade.
- ❖ Utilisez un « foreach » pour afficher chaque élément de votre tableau dans votre vue.
- ❖ Cherchez, sur la doc Laravel Blade, comment ne pas afficher le première et le dernier élément d'un tableau lors d'une itération avec un for/foreach Blade.
- ❖ Convertissez le tableau en JSON directement dans la vue à l'aide d'une commande Blade. Ensuite, créez une variable en JS (dans des balise script) qui recueillera le tableau.



# Blade - Layout & include

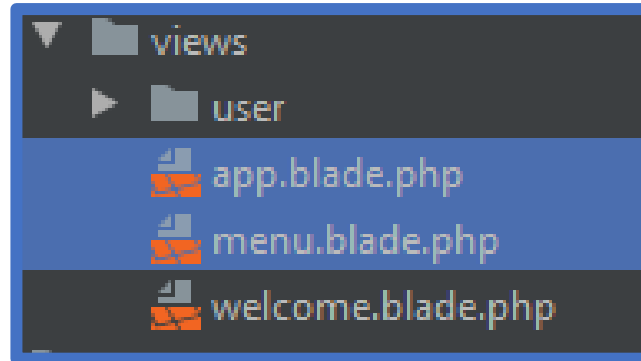


Chapitre 2





## La gestion pyramidale 1/2



- Un **layout contenant le minima HTML** (souvent appelé « app.blade.php »). Il s'agit du parent des autres vues. L'ajouter d'autres layouts est possible.
- 2 directives Blade pour les layouts (A ne pas appeler avec des contrôleurs ou des routes) :
  - « **@yield('STRING')** » pour signifier qu'il est possible d'ajouter du code d'autres vues Blade.
  - « **@include('STRING', ['var' => \$var])** » pour inclure l'intégralité d'une autre vue Blade (ex: un menu).

➔ [Voir la doc](#)

```
app.blade.php x
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5
6     <!-- Style Css -->
7     @yield('style')
8
9     <!-- Title -->
10    <title>@yield('title')</title>
11 </head>
12 <body>
13     <!-- Main navigation bar -->
14     @include('menu', [
15         'var' => $var
16     ])
17
18     <!-- Content -->
19     <main>
20         @yield('content')
21     </main>
22
23     <!-- Script Body -->
24     @yield('script')
25 </body>
26 </html>
```





## La gestion pyramidale 2/2

```
menu.blade.php
1 <nav>
2   <a href="@route('welcome')">Welcome</a>
3
4   @auth
5     <a href="@route('user.profile')">Profile</a>
6     <a href="@route('logout')">Logout</a>
7   @endauth
8
9   @guest
10    <a href="@route('login')">Profile</a>
11  @endguest
12 </nav>
```

- « **@extends('STRING')** » pour choisir le layout à utiliser dans la vue enfant.
- « **@section('STRING') ... @endsection** » pour définir le code à insérer à la place du « @yield » correspondant dans le layout parent.
- L'essentiel en 4 commandes Blade :  
→ **@yield()**, **@include()**, **@extends()** et **@section()**

```
profile.blade.php
1 {{-- App layout --}}
2 @extends('app')
3
4 @section('style')
5   <link rel="stylesheet" href="...">
6 @endsection
7
8 @section('title')
9   Title page !
10 @endsection
11
12 @section('content')
13   Body page !
14 @endsection
15
16 @section('script')
17   <script type="text/javascript" src="..."></script>
18 @endsection
19
```



Qu'est ce que Laravel Blade ?

## Place à la pratique

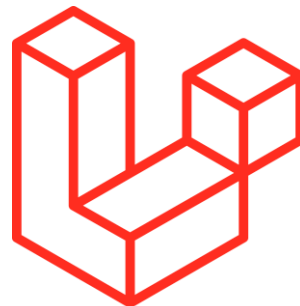
- ❖ Créez une vue Blade « **app** » et y mettre le minimum HTML.
- ❖ Ajoutez les @yields « **style, title, content** et **script** » et un @include « **menu** » au bonne endroit.
- ❖ Créez une vue Blade « **menu** » et y mettre un lien vers les routes « **welcome, pokemons.index** et **pokemons.create** ». Créez les routes et le contrôleur si besoin.
- ❖ Modifiez votre vue Blade « **pokemons.index** », « **pokemons.create** » et « **pokemons.edit** » en y ajoutant l'extension « **app** » et les sections correspondantes.



# Les assets frontend



## Chapitre 3



# Laravel Mix - Présentation

- Le meilleur ami de Laravel c'est NodeJs
- NodeJs 18+
- Npm 8+

```
#> npm install
```

```
#> npm update
```

→ Installation des dépendances nodeJs (à faire uniquement après avoir préparé votre package.json)

```
#> npm run dev
```

```
#> npm run prod
```

→ Les 2 commandes pour compiler les assets mix en éléments statiques dans public/

- Vite.js remplace Laravel Mix depuis Laravel 9 mais il fonctionne différemment.
- Vite est plus complet et plus automatisé mais moins répandue.



## Installation: package.json

```
{
  "private": true,
  "scripts": {
    "dev": "npm run development",
    "development": "mix",
    "watch": "mix watch",
    "watch-poll": "mix watch -- --watch-options-poll=1000",
    "hot": "mix watch --hot",
    "prod": "npm run production",
    "production": "mix --production"
  },
  "devDependencies": {
    "axios": "^1.6.4",
    "laravel-mix": "^6.0.49",
    "resolve-url-loader": "^5.0.0",
    "sass": "^1.71.0",
    "sass-loader": "^12.6.0"
  }
}
```

1. Remplacez le contenu de votre « package.json » par le contenu ci-contre.
2. Lancez la commande « npm install » depuis la racine du dossier Laravel (la où se trouve le package.json).
3. Créez un fichier « webpack.mix.js » à la racine de l'application.



# Installation: Exemple de webpack.mix.js

- Exemple complet d'un webpack avec Laravel Mix

```
// webpack.mix.js

let mix = require('laravel-mix');
mix.disableSuccessNotifications();

mix.js('resources/js/app.js', 'public/assets/js/')
  .sass('resources/sass/app.scss', 'public/assets/css/');

if (mix.inProduction()) {
  mix.version();
}
```



# Laravel Mix

```
webpack.mix.js x
JS
1  const mix = require('laravel-mix');
2
3  mix.js('resources/js/app.js', 'public/js')
4      .sass('resources/sass/app.scss', 'public/css')
5      .version();
6
7  mix.copy('resources/js/scroll.js', 'public/js/scroll.js');
8
9  mix.styles([
10     'public/css/vendor/normalize.css',
11     'public/css/vendor/videojs.css'
12 ], 'public/css/all.css');
13
14 mix.scripts([
15     'node_modules/pickerdate/src/pickerdate.js',
16     'node_modules/pickerdate/src/pickerdate-fr.js',
17 ], 'public/js/pickerdate.js');
```

- Copie le css ou js
- Compile le scss en css et minifie le css et js.
- Assemble des fichiers css ou js en un
- Optimisé pour un mode de production (« npm run prod »)
- Forte compatibilité
- Versionnage par « npm run »





## Sass / CSS

```
// Sass
mix.sass('resources/sass/app.scss', 'public/css')
  .version();

// Less
mix.less('resources/less/style.less', 'public/css')
  .version();

// CSS Plain
mix.styles([
  'public/css/vendor/normalize.css',
  'public/css/vendor/videojs.css'
], 'public/css/all.css');

// Copy
mix.copy('resources/css/scroll.css', 'public/css/scroll.css');
```

```
<link rel="stylesheet" href="/css/app.css">
<link rel="stylesheet" href="{ { mix('/css/app.css') } }">
```

- Sass
- Less
- Plain CSS (CSS standard)
- ...
- Génération d'url en option

→ mix() uniquement si le versionning est utilisé



## JS

```
// JS
mix.js('resources/js/app.js', 'public/js');
mix.scripts([
    'node_modules/pickerdate/src/pickerdate.js',
    'node_modules/pickerdate/src/pickerdate-fr.js',
], 'public/js/pickerdate.js');
```

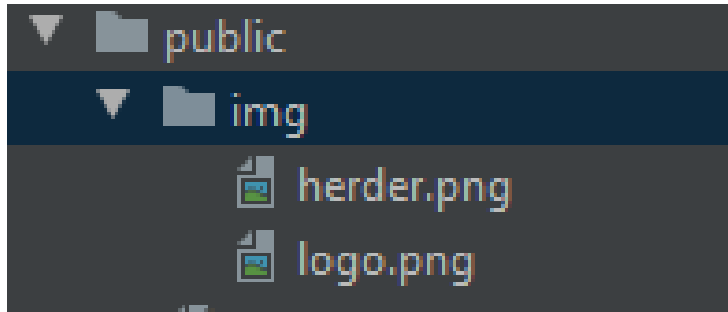
- Vanilla JS
- ReactJs

```
<script type="text/javascript" src="/js/app.js"></script>
<script type="text/javascript" src="{ mix('/js/app.js') }"></script>
```

→ « mix() » uniquement si le versionning est utilisé



## Assets, images et fichiers publics



- Dans « public/... »
- Pas de structure particulière. En général des dossiers : « css/... », « js/... » et « img/... » à placer dans un dossier « /public/assets/ »
- Les fichiers privés se gèrent avec le Storage de Laravel

```
mix.copyDirectory('resources/img', 'public/img');
```

→ Copie tous les dossiers et fichiers d'un dossier à un autre. A utiliser plutôt pour les images entre « /resources/ » et « /public/ ». Attention à ne pas se tromper sous peine de copie-coller des dossiers et fichiers n'importe où et d'endommager votre application Laravel.



## Pratiquer c'est mieux

- ❖ Installez bootstrapcss avec « `npm install bootstrap` ».
- ❖ Créez un fichier « `resources/sass/app.scss` » et importez y « `bootstrap` ». Il s'agit d'une méthode propre au CSS et SASS.
- ❖ Dans « `webpack.mix.js` », ajoutez la bonne méthode pour importer votre « `app.scss` » et votre « `app.js` ». Voir le cours.
- ❖ Ajoutez ensuite une importation de ces derniers dans votre layout `app.blade.php` (N'oubliez pas de les compiler). Les importations doivent venir de fichiers présent dans le dossier public et non des ressources.
- ❖ Vérifiez que l'importation fonctionne bien.



```
17 string sInput;  
18 int iLength, iN;  
19 double dblTemp;  
20 bool again = true;  
21  
22 while (again) {  
23     iN = -1;  
24     again = false;  
25     getline(cin, sInput);  
26     system("cls");  
27     stringstream(sInput) >> dblTemp;  
28     iLength = sInput.length();  
29     if (iLength < 4) {  
30         again = true;  
31         continue;  
32     } else if (sInput[iLength - 3] != '.') {  
33         again = true;  
34         continue;  
35     } while (++iN < iLength) {  
36         if (isdigit(sInput[iN])) {  
37             continue;  
38         } else if (iN == (iLength - 3)) {  
39             continue;  
40         }  
41     }  
42     // ...  
43 }
```

Fin du module

