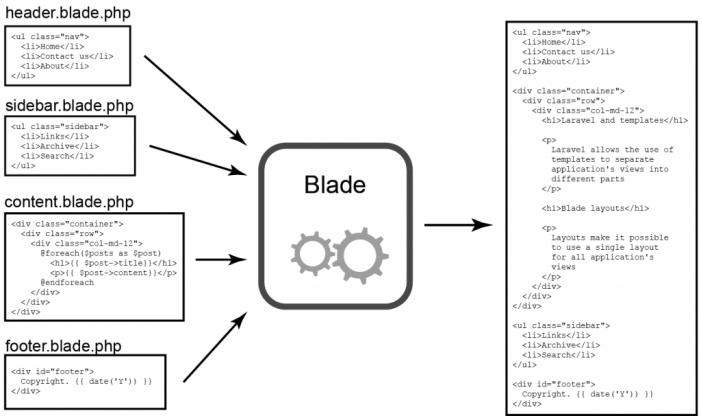
Laravel Blade



Laravel Blade est le moteur de templates intégré de Laravel. Il permet de composer des vues en utilisant une syntaxe simple et intuitive tout en profitant de la puissance de PHP. Voici un guide détaillé sur l'utilisation de Blade dans Laravel.

Syntaxe de Base

default.blade.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
  ...
  <title>@yield('title')</title>
  ...
  <link href="..." type="text/css" rel="stylesheet" />
  ...
  @yield('custom_css')
  ...
</head>
<body>
  ...
  @yield('content')
  ...
  <script src="..."></script>
  ...
  @yield('scripts')
  ...
</body>
</html>
```

page1.blade.php

```
@section('title')
  Titre de la page 1
@endsection

@section('content')
  <div> ....</div>
@endsection
```

page2.blade.php

```
@section('title')
  Titre de la page 2
@endsection

@section('custom_css')
  <link href=.../>
@endsection

@section('content')
  <div> ....</div>
@endsection
```

Affichage de Données

Pour afficher des données, utilisez les accolades doubles `{{ }}` :

```
{{ $variable }}
```

Pour échapper les données (prévenir les attaques XSS) :

```
{{{ $variable }}}
```

Pour afficher des données non échappées (si vous êtes sûr qu'elles sont sécurisées) :

```
{!! $variable !!}
```

Structures de Contrôle

Conditionnelles

```
@if ($condition)
    <!-- contenu -->
@elseif ($anotherCondition)
    <!-- contenu -->
@else
    <!-- contenu -->
@endif
```

Boucles

For Loop

```
@for ($i = 0; $i < 10; $i++)
    <p>{{ $i }}</p>
@endfor
```

Foreach Loop

```
@foreach ($items as $item)
    <p>{{ $item }}</p>
@endforeach
```

Forelse Loop (avec gestion des listes vides)

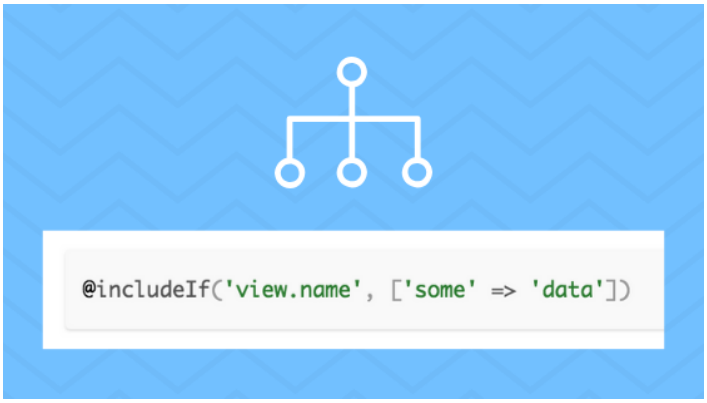
```
@forelse ($items as $item)
    <p>{{ $item }}</p>
@empty
```

```
    <p>Aucun élément trouvé.</p>
@endforelse
```

While Loop

```
@while ($condition)
    <p>La condition est vraie.</p>
@endwhile
```

Inclusions



Pour inclure une vue à l'intérieur d'une autre vue :

```
@include('view.name')
```

Vous pouvez aussi passer des données à la vue incluse :

```
@include('view.name', ['data' => $value])
```

Sections et Layouts

Blade permet de définir des layouts et d'étendre ces layouts dans différentes vues.

Définir un Layout (master.blade.php)

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title')</title>
</head>
<body>
    @yield('content')
</body>
</html>
```

Étendre un Layout (child.blade.php)

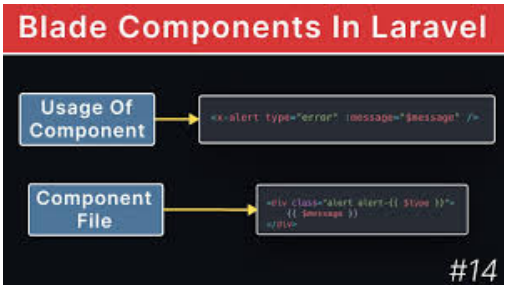
```
@extends('layouts.master')

@section('title', 'Page Title')

@section('content')
    <p>Ce contenu remplacera la section content du layout master.</p>
@endsection
```

Components

Blade permet de créer des composants réutilisables.



Définir un Composant (resources/views/components/alert.blade.php)

```
<div class="alert alert-{{ $type }}">
    {{ $slot }}
</div>
```

Utiliser un Composant

```
<x-alert type="error">
    Ceci est un message d'erreur.
</x-alert>
```

Slots

Les slots permettent de passer des sections de contenu à un composant.

Composant avec Slots (resources/views/components/layout.blade.php)

```
<div class="container">
    <header>{{ $header }}</header>
    <div class="content">{{ $slot }}</div>
</div>
```

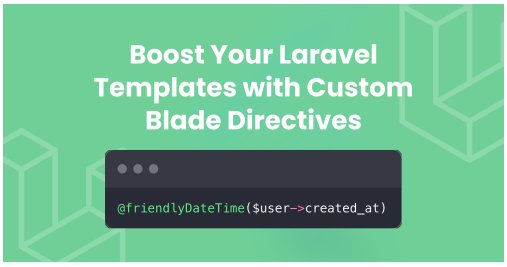
Utilisation du Composant avec Slots

```
<x-layout>
    <x-slot name="header">
        Ceci est l'en-tête.
    </x-slot>

    Ceci est le contenu principal.
</x-layout>
```

Directives Personnalisées

Vous pouvez définir vos propres directives Blade.



Définir une Directive Personnalisée (dans AppServiceProvider par exemple)

```
use Illuminate\Support\Facades\Blade;

Blade::directive('datetime', function ($expression) {
    return "<?php echo ($expression)->format('m/d/Y H:i'); ?>";
});
```

Utiliser une Directive Personnalisée

```
@datetime($date)
```

Conclusion

Laravel Blade est un moteur de template puissant et flexible, conçu pour rendre le développement d'applications web rapide et agréable. En utilisant les fonctionnalités de Blade, vous pouvez créer des vues dynamiques et bien structurées, tout en profitant d'une syntaxe propre et lisible.