

# Bloc Laravel • chapitre 9

# Views

Les lignes communes du site s'écrivent une seule fois, dans un fichier que toutes les pages réutilisent.



# Trois pages qui se ressemblent

```
1  ...
2  <title>Infos pratiques • Les Nuits du Parc</title>
3  ...
4  <header>
5      <p>Les Nuits du Parc</p>
6  ...
7  <main>
8      <h1>Infos pratiques</h1>
9      <p>Le festival a lieu au parc communal, du vendredi au dimanche.</p>
10 </main>
```



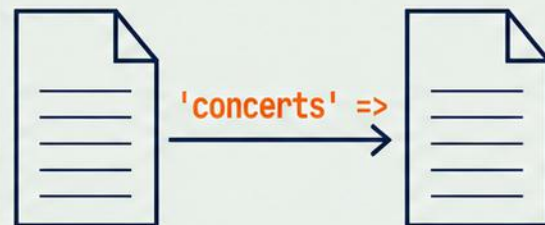
Seuls le **<title>** et le contenu de **<main>** changeraient d'une page à l'autre.

# Le contrôleur passe un tableau à la vue

Le contrôleur renvoie `view('concerts.index', ['concerts' => $this->concerts])`.

```
1 <h1>Programme</h1>
2 <ul>
3     @foreach ($concerts as $concert)
4         <li>{{ $concert['artist'] }}, {{ $concert['stage'] }}, {{ $concert['time'] }}</li>
5     @endforeach
6 </ul>
```

```
$ curl -s http://premier-test-laravel.test/concerts
<h1>Programme</h1>
<ul>
    <li>Duo Mistral, Chapiteau, 18 h</li>
    <li>L'Orchestre du Lac, Grande scène, 20 h</li>
...
```

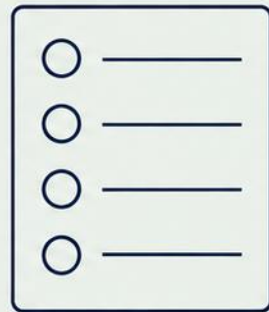


# Les directives de tous les jours

Une **directive** est une instruction Blade qui commence par @, comme @foreach.

```
1 @if ($concert['duration'] >= 90)
2     <p>Long concert : {{ $concert['duration'] }} minutes, avec entracte.</p>
3 ...
4 @endif
5 ...
6 @forelse ($concert['guests'] as $guest)
7     <li>{{ $guest }}</li>
8     @empty
9         <li>Aucun invité annoncé</li>
10    @endforelse
```

```
$ curl -s http://premier-test-laravel.test/concerts/2
...
<p>Long concert : 90 minutes, avec entracte.</p>
...
    <li>Ana Lopes</li>
```



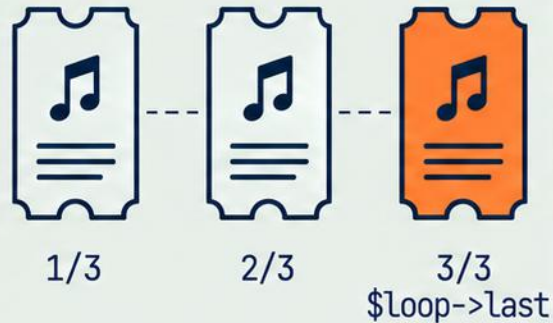
0 invité

# \$loop décrit le tour en cours

Dans une boucle @foreach, Blade crée une variable **\$loop**, un objet qui décrit le tour en cours.

```
1 @foreach ($concerts as $id => $concert)
2     <li>
3         {{ $loop->iteration }}/{{ $loop->count }}
4         <a href="{{ route('concerts.show', $id) }}">{{ $concert['artist'] }}</a>
5         @if ($loop->last)
6             (clôture)
7         @endif
8     </li>
9 @endforeach
```

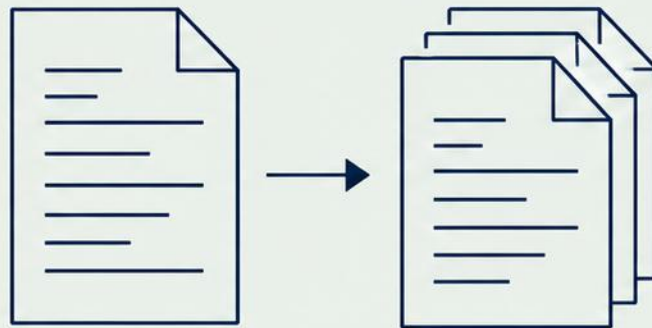
```
$ curl -s http://premier-test-laravel.test/concerts
...
    3/3
    <a href="http://premier-test-laravel.test/concerts/3">Orage Mécanique</a>
        (clôture)
...
```



# Blade traduit le template en PHP

```
1 <p><?php echo e($concert['description']); ?></p>  
2 <p><?php echo $concert['description']; ?></p>
```

concerts/show.blade.php      storage/framework/views/



`{{ }}` passe par la fonction `e()`, `{!! !!` non.

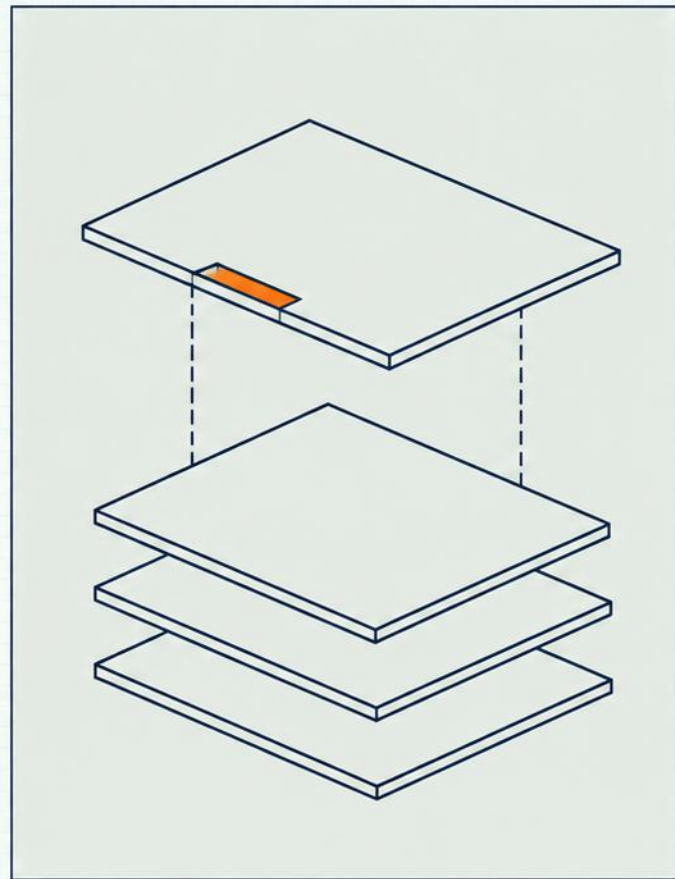
**N'utilisez `{!! !!` que pour du HTML que vous avez écrit vous-même**

# Un composant s'écrit une fois et sert partout

Un composant Blade est un morceau de template que vous écrivez une fois et réutilisez dans plusieurs vues.

```
1 <div class="alert">
2     {{ $slot }}
3 </div>
```

Le texte écrit entre les balises `<x-alert>` arrive dans `{{ $slot }}`.



# @props déclare les données du composant

```
1 @props(['type' => 'info'])
2
3 <div class="alert alert-{{ $type }}">
4     {{ $slot }}
5 </div>
```

```
1 <div class="alert alert-danger">
2     Le concert de 22 h est complet.
3 </div>
```



Avec type="danger", la prop type vaut danger. Sans l'attribut, elle vaut 'info'.

# Pour passer la valeur d'une expression PHP, écrivez deux-points devant le nom de l'attribut

```
1 <x-concert-card artist="$concert['artist']" time="$concert['time']" />
```

```
<h2>$concert[&#039;artist&#039;]</h2>
```



Corrigé : :artist="\$concert['artist']" affiche Duo Mistral.

# Un second contenu, dans <x-slot:...>

Un slot nommé est un contenu supplémentaire que la vue transmet au composant dans une balise <x-slot:...>.

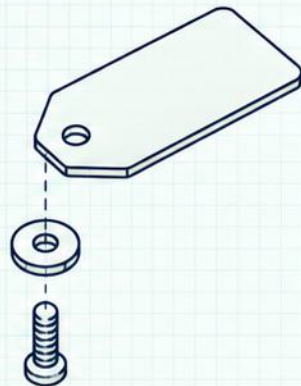


```
1 <x-concert-card :concert="$concert">
2   @if ($concert['duration'] >= 90)
3     <x-slot:note>Entracte de 15 minutes</x-slot:note>
4   @endif
5 </x-concert-card>
```

Le composant le lit dans \$note, entouré de @isset.

# \$attributes reçoit ce que @props ne déclare pas

```
1 @props(['type' => 'info'])
2
3 <div {{ $attributes->merge(['class' => 'alert alert-' . $type]) }}>
4     {{ $slot }}
5 </div>
```



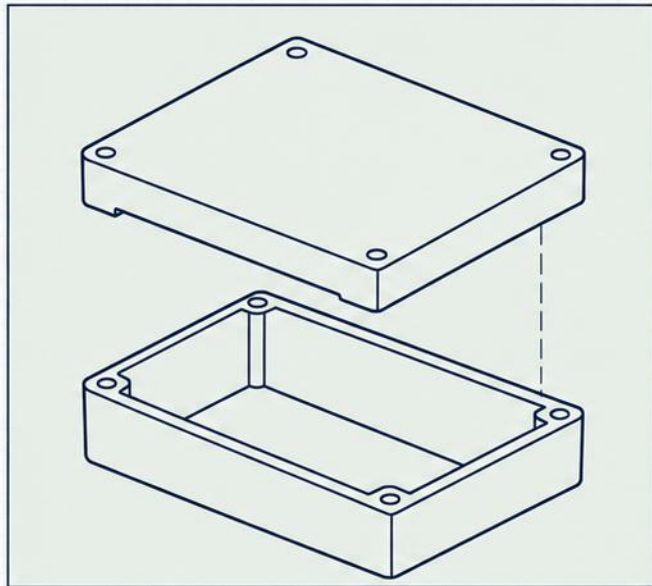
```
<div class="alert alert-danger top" id="sold-out">
    Le concert de 22 h est complet.
</div>
```

**Déclarez avec @props chaque donnée de votre composant.**

# Le layout du site est un composant

Un layout est le template qui contient ce que les pages d'un site partagent : le <head>, l'en-tête, le menu et le pied de page.

```
1  @props(['title' => 'Festival'])
2  ...
3      <title>{{ $title }} · Les Nuits du Parc</title>
4  ...
5      <a href="{{ route('concerts.index') }}">Programme</a>
6  ...
7      <main>
8          {{ $slot }}
9      </main>
10  ...
```

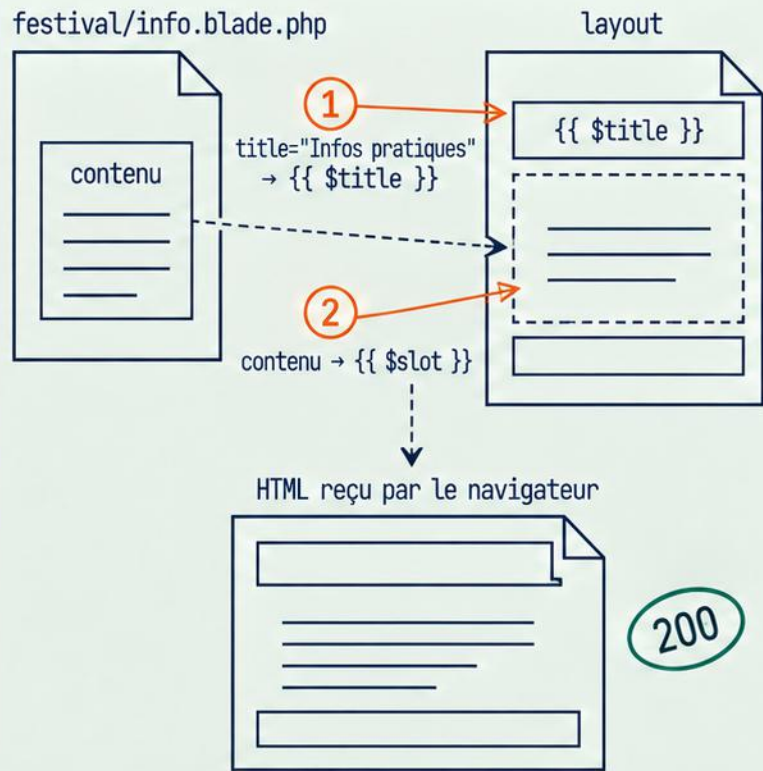


```
$ php artisan make:component layouts.app --view
```

**INFO** View [resources/views/components/layouts/app.blade.php] created successfully.

# La page ne contient que ce qui lui est propre

```
1 <x-layouts.app title="Infos pratiques">
2   <h1>Infos pratiques</h1>
3   <x-alert>Les portes ouvrent à 17 h 30.</x-alert>
4   <x-alert type="danger" class="top" id="sold-out">
      Le concert de 22 h est complet.</x-alert>
5   <p>Le festival a lieu au parc communal, du vendredi
      au dimanche.</p>
6 </x-layouts.app>
```



Pour renommer le festival, vous modifiez trois lignes d'un seul fichier.

# L'ancienne façon : quatre directives

|   |                                                   |
|---|---------------------------------------------------|
| 1 | <code>@extends('layouts.base')</code>             |
| 2 |                                                   |
| 3 | <code>@section('title', 'Infos pratiques')</code> |
| 4 |                                                   |
| 5 | <code>@section('content')</code>                  |
| 6 | <code>&lt;h1&gt;Infos pratiques&lt;/h1&gt;</code> |
| 7 | <code>...</code>                                  |
| 8 | <code>@endsection</code>                          |

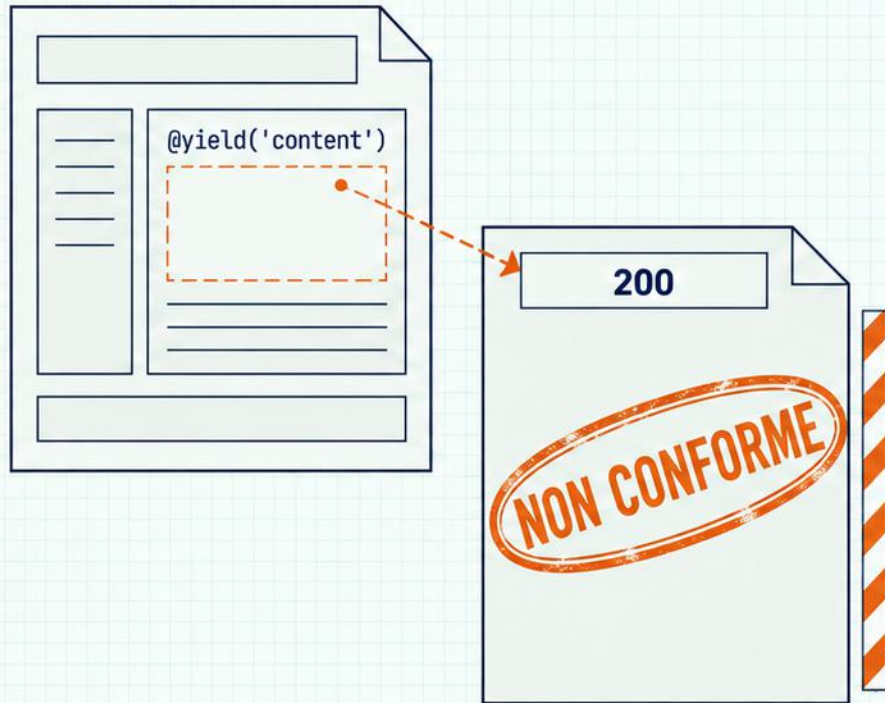
| Directive                                | Rôle                         |
|------------------------------------------|------------------------------|
| <code>@extends('layouts.base')</code>    | le layout de la page         |
| <code>@section('content')</code>         | définit un bloc de contenu   |
| <code>@yield('content')</code>           | place le bloc dans le layout |
| <code>@include('partials.header')</code> | insère un autre template     |

# Lisez @extends dans le code existant. Écrivez des composants dans le vôtre.

Avec `@section('contenu')` au lieu de `content`,  
la page arrive vide, sans aucun message :

```
1 $ curl -s -i http://premier-test-laravel.test/ancien/infos
2 HTTP/1.1 200 OK
3 ...
4     <main>
5         </main>
```

Avec un composant, une faute de nom provoque une erreur.



# Blade et Vue.js côte à côte

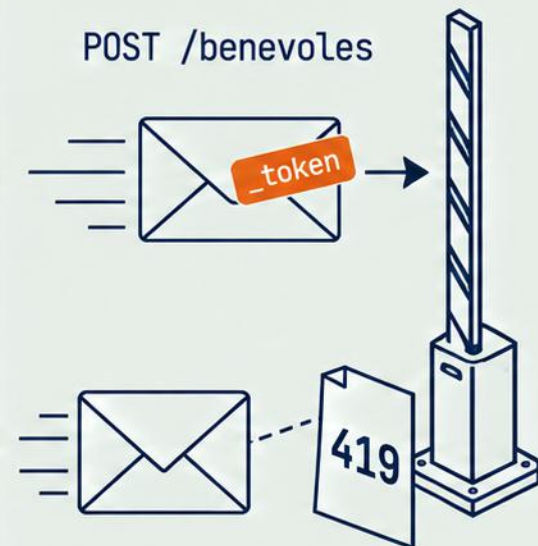
| Idée                           | Blade, composants                                                                                            | Vue.js                                                                           |
|--------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| Déclarer les données attendues | <code>@props(['concert'])</code>                                                                             | <code>defineProps(['concert'])</code>                                            |
| Passer une expression          | <code>:concert="\$concert" (PHP)</code>                                                                      | <code>:concert="concert" (JavaScript)</code>                                     |
| Zone de contenu principal      | <code>{{ \$slot }}</code>  | <code>&lt;slot /&gt;</code>                                                      |
| Zone nommée                    | <code>{{ \$title }}</code> et <code>&lt;x-slot:title&gt;</code>                                              | <code>&lt;slot name="title" /&gt;</code> et <code>&lt;template #title&gt;</code> |

Blade exécute un composant une seule fois, sur le serveur. Un composant Vue tourne dans le navigateur.

# @csrf ajoute le jeton que Laravel attend

```
1 <form method="POST" action="{{ route('volunteers.store') }}">
2   @csrf
3   <label>Prénom <input type="text" name="first_name"></label>
```

```
1 $ curl -s http://premier-test-laravel.test/benevoles | grep _token
2 <input type="hidden" name="_token" value="6HjTJozqH7XAcXVJCTyVptQjZ9LRXU2dZV2gkYri"
  autocomplete="off">...
```

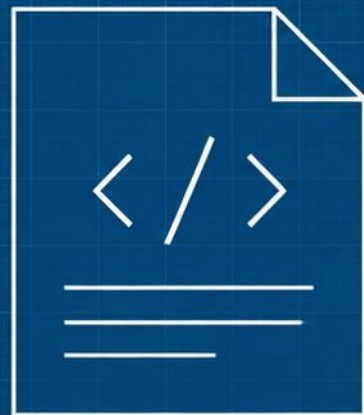


Sans @csrf, le navigateur affiche Page Expired, code 419.  
**Écrivez @csrf dans chaque formulaire POST.**

# À vous : le site d'Octet

115 minutes, en séance ou chez vous.

- Ex. 1 à 3 : directives.
- Ex. 4 à 6 : composants.
- Ex. 7 et 8 : layouts et @csrf.
- Ex. 9 : diagnostiquer, sur papier.



components/layouts/company.blade.php

