

# Corso Laravel

## View & Components

# Le views

La componente view ( o logica di presentazione) è la parte in cui l'utente finale visualizza i dati manipolati e interpretati dal controller.

In pochissime parole: la view è il codice html caricato dal browser.

# Blade

Blade è il template engine predefinito di Laravel.

Un file blade non è altro che una view scritta in php. Dovrai aggiungerci soltanto una pre-estensione

`view.blade.php`

A differenza degli altri template engine, Blade non ha alcuna restrizione sul PHP. Potrai scrivere direttamente porzioni di codice nativo senza problemi.

Di default tutte le views dovrai inserirle nel percorso `resources/views`.

Da qui, verranno inizialmente compilate e, successivamente, memorizzate nella cache di sistema (`storage/framework`). Le views non verranno processate e compilate ogni volta, ma semplicemente caricate e lette in modo statico.

# Richiamare una View

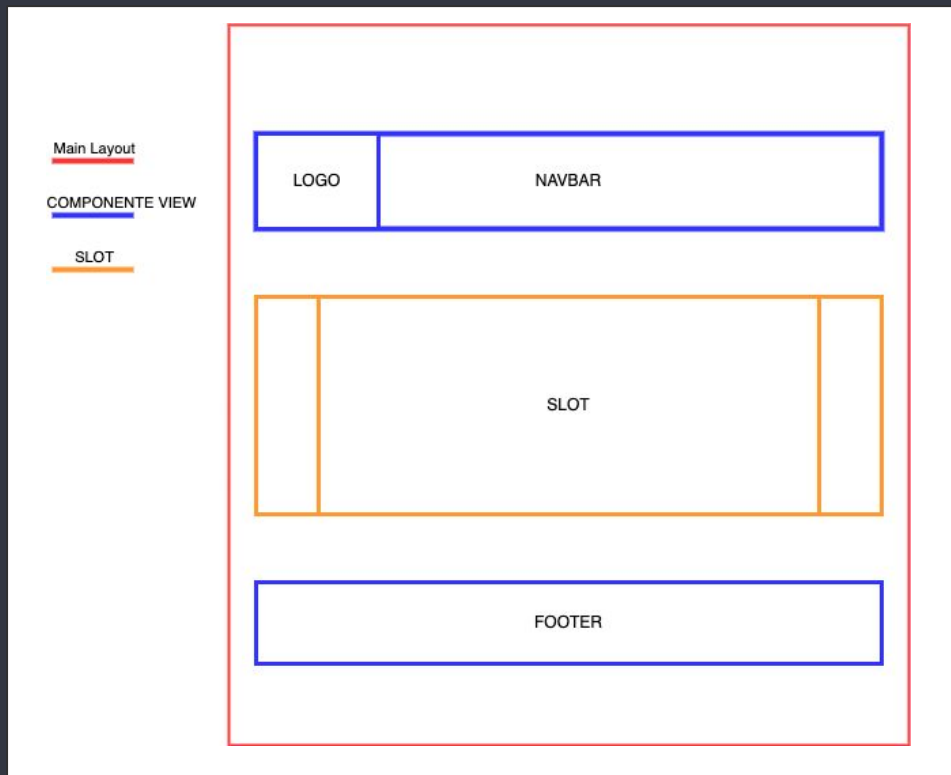
```
<?php
namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
use App\User;

class UserController extends Controller
{
    public function profile($id){
        return view('welcome');
    }
}
```

# Come passare i dati ad una view

```
public function homepage(){  
    $data = [  
        'name' => 'Francesco',  
        'country' => 'ITA'  
    ];  
    return view('homepage', ['data' => $data]);  
    return view('homepage')->with('data', $data);  
    return view('homepage', compact('data'));  
}
```

# A cosa serve un Layout




# Strutturare un Layout

Il comando appena lanciato andrà a creare la view in

`resources/views/components/main.blade.php`

E andremo ad inserire la variabile `$slot` in quanto quello sarà il nostro segnaposto che, come un portale (Avete mai visto la serie TV Stargate?), stamperà il contenuto ereditato.



```
<body>
<main>
    {{$slot}}
</main>
<footer>Francesco Mansi</footer>
</body>
```

# Strutturare un Layout

Ora, non ci resta che creare tutte le pagine che andranno ad estendere quel layout. Come? In questo modo:



```
<x-main-layout>  
  
<h1>Ciao</h1>  
  
</x-main-layout>
```



# Strutturare un Layout

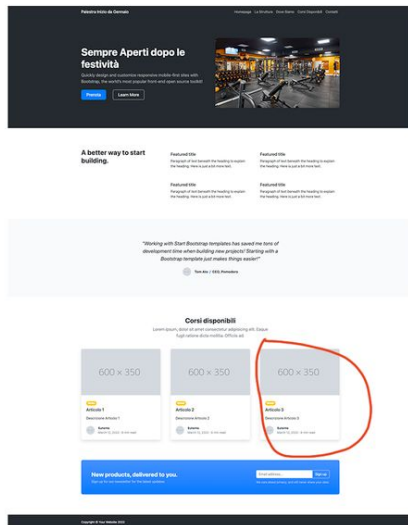
Il risultato finale sarà questo:



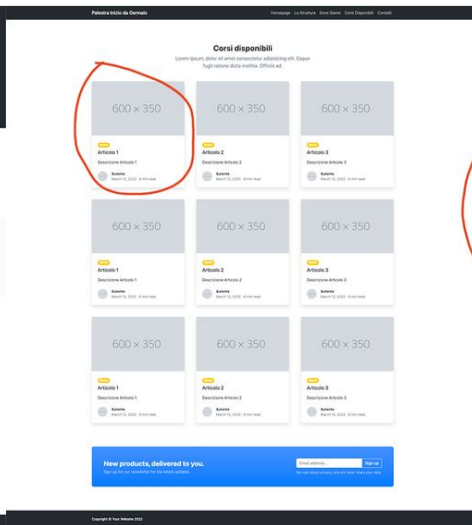
```
//File finale - HTML
<body>
<main>
  <h1>Ciao</h1>
</main>
  <footer> Francesco Mansi </footer>
</body>
```

# A cosa serve un componente

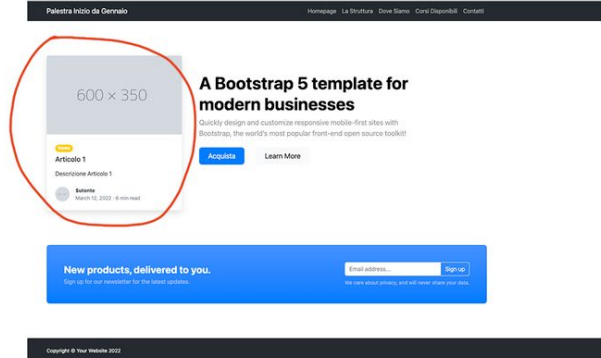
## Homepage



## Elenco Corsi



## Dettaglio Corso



3 pagine diverse, stesso componente

# Strutturare un Layout

Laravel questa volta copia a piene mani dai linguaggi frontend come Vue e React, appropriandosi della logica dei componenti.

Sempre rispettando il principio della segregazione delle interfacce e di singola responsabilità (SOLID) utilizzare una struttura a componenti ti permetterà di scrivere sempre meno codice e di qualità.



```
php artisan make:component Main --view
```

# Passare i dati nel componente



```
<x-card :data="$variabile_php" :altro="'Stringa da inserire'"/>
```

# Passare i dati nel componente

## Short Syntax



```
<x-profile :$userId :$name />
```

```
{{-- Equivale --}}
```

```
<x-profile :user-id="$userId" :name="$name" />
```

# Slot principale

Ogni componente, di default, ha uno \$slot disponibile:

```
<body>
  <main>
    {{$slot}}
  </main>
</body>
```

```
<x-app-layout>
  Contenuto da Stampare
</x-app-layout>
```

# Slot Custom

Ogni componente, di default, ha uno \$slot disponibile:

```
<!DOCTYPE html>
<html lang="{ str_replace('_', '-', app()→getLocale()) }">
  <head>
    {{ $title }}
  </head>
  <body >
    <div>
      <main>
        {{ $slot }}
      </main>
    </div>
    {{ $error }}
  </body>
</html>
```

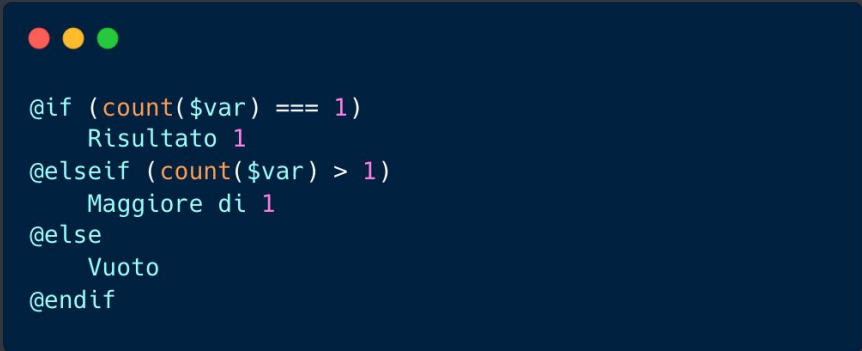
```
<x-app-layout>
  <x-slot name="header">
    Titolo
  </x-slot>

  <x-slot:error>
    Server Error
  </x-slot>
</x-app-layout>
```

# EXTRA:Comandi Blade

IF condizionale con @if, @else, @endif

Creare una struttura di controllo IF risulterà molto semplice. Ti basterà anteporre la chiocciola al metodo di controllo per utilizzarlo immediatamente.



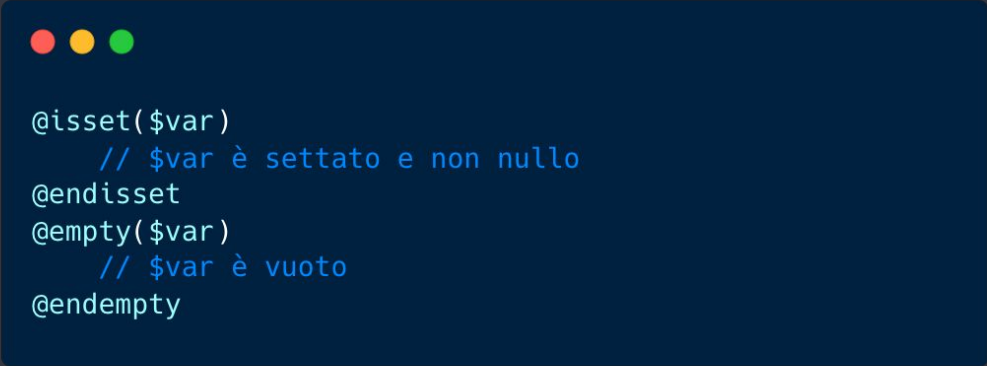
```
@if (count($var) === 1)
    Risultato 1
@endif
@elseif (count($var) > 1)
    Maggiore di 1
@else
    Vuoto
@endif
```



# EXTRA:Comandi Blade

Null o vuoto con @isset o @empty

Anche i famosissimi metodi PHP sono disponibili in Blade, potrai richiamarli in questo modo:



```
@isset($var)
    // $var è settato e non nullo
@endisset
@empty($var)
    // $var è vuoto
@endempty
```

# EXTRA:Comandi Blade

Gestire tutti i casi con @switch, @case, @break, @default

Potresti voler confrontare una variabile su più condizioni, valori diversi e con risultati diversi. La struttura switch fa al caso tuo, ecco come implementarla:

```
@switch($age)
    @case($age > 0 && $age < 19)
        //Minorenne
    @break
    @case($age > 18 && $age < 100)
        //Maggiorenne
    @break
    @default
@endswitch
```

# EXTRA:Comandi Blade

Uno sguardo ai Loop

Blade semplifica anche le strutture classiche di PHP, come i cicli foreach, for e while.

Eccone un'assaggio:

```
@for ($i = 0; $i < 10; $i++)  
    Il valore è {{ $i }}  
@endfor  
  
@foreach ($users as $user)  
    <p>Ciao, {{ $user->name }}</p>  
@endforeach  
  
@forelse ($users as $user)  
    <li>{{ $user->name }}</li>  
@empty  
    <p>Nessun utente</p>  
@endforelse  
  
@while (true)  
    <p>Verso l'infinito e oltre!</p>  
@endwhile
```

# EXTRA:Comandi Blade

Dentro un ciclo foreach potrai accedere a tantissimi metodi utili:

- ``$loop->index``: L'indice corrente (inizia da 0);
- ``$loop->iteration``: L'iterazione del loop corrente (inizia da 1);
- ``$loop->remaining``: Quanti indici mancano alla fine;
- ``$loop->count``: Il numero totale degli elementi;
- ``$loop->first``: Se questa è la prima iterazione del ciclo;
- ``$loop->last``: Se questa è la prima iterazione del ciclo;
- ``$loop->even``: Se questa è un'iterazione pari;
- ``$loop->odd``: Se questa è un'iterazione dispari;
- ``$loop->depth``: Il livello di annidamento del loop corrente.
- ``$loop->parent``: Quando si trova in un ciclo nidificato, la variabile del ciclo del genitore.

```
@foreach ($users as $user)

    @if ($user->type == 1)
        @continue
    @endif
    <li>{{ $user->name }}</li>
    @if ($user->number == 5)
        @break
    @endif
@endforeach
```