## TUJUAN PRAKTIKUM

1. Mahasiswa mampu memahami konsep view dalam framework.
2. Mahasiswa mampu memahami konsep blade dalam framework.
3. Mahasiswa mampu membuat view dan blade.
4. Mahasiswa mampu memahami kontrol struktur dengan blade template.
5. Mahasiswa mampu mengetahui master template blade (layout).

## TEORI SINGKAT

View merupakan bagian yang menampilkan informasi untuk disampaikan kepada users. View terdiri dari script HTML dengan bantuan CSS dan javascript. Sesuai dengan aturan konsep MVC, di dalam view tidak boleh ada script logika maupun script untuk mengakses database. Di dalam laravel view diletekan di dalam folder **resources/views/.**

Sedangkan blade template merupakan engine yang disediakan laravel untuk memudahkan developer dalam menampilkan data pada view. Dengan blade tempalte ini kita tidak perlu lagi menggunakan **<?php echo $data ?>** untuk menampilkan variabel data pada view. Untuk menggunakan blade tempalete, file view harus disimpan dengan akhiran **.blade.php** dan disimpan pada folder **resourcess/views.**

## PRAKTIKUM

### MEMBUAT VIEW

Untuk mempraktekan pembuatan view, terlebih dahulu kita membuat sebuah route untuk mengakses view tersebut. Route yang dibuat akan langsung mengarah ke controller **mhsController.php** yang telah dibuat pada latihan sebelumnya. Route yang dibuat adalah seperti pada sintak dibawah:

```
Route::get('/mhs', [mhsController::class, 'index'])->name('mhs');
```

Pada route diatas, jika pada browser dituliskan alamat http://**localhost:8000/mhs** maka kita akan diarahkan ke class mhsController ke dalam fungsi index. Selanjutnya bukalah file **mhsController.php** yang ada di dalam folder **app\Http\Controller** dan kemudian modifikasi script nya menjadi seperti berikut:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class mhsController extends Controller
{
    public function index(){
        $mhs = "Nadia";
        return view('mhs/index',compact('mhs'));
    }
}
```

Di dalam fungsi index() yang telah dibuat kita mendefinisikan satu variabel bernama **mhs** yang langsung kita ikut sertakan lempar bersama dengan return view yang dimana view yang akan dicari berada pada folder projek Laravel yaitu **websaya/resources/views/mhs/** dengan nama **index.blade.php.** karena view tersebut belum dibuat maka buatlah satu file baru pada folder **websaya/resources/views/mhs/** dan beri nama **index.blade.php** dan isikan skript sebagai berikut:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>WebSaya.Com</title>
    </head>
    <body>
        <h1>Selamat Datang {{ $mhs }} di WebSaya.Com</h1>
    </body>
</html>
```

Pada script diatas kita menampilkan value dari variabel mhs dengan hanya menggunakan tanda {{ dan ditutup dengan tanda }}. Fitur yang memungkinkan untuk menampilkan data dengan cara tersebut yaitu **blade template** yang akan dibahas pada materi selanjutnya. Dan jika kita membuka link http://**localhost:8000/mhs** maka akan muncul halaman seperti berikut:


## MENAMPILKAN DATA DENGAN BLADE TEMPLATE

Biasanya untuk menampilkan variabel PHP pada HTML kita menggunakan script **<?php echo $data ?>**, dengan blade template kita cukup menulisnya dengan script **{{ $data }}.** Cara ini juga sudah dilengkapi dengan pencegahan XXS attacks ( hacker yang memanfaatkan javascript untuk mencuri informasi ). Untuk melihat contoh dari penggunaan data tersebut dapat dilihat pada contoh sebelumnya. Pada kasus lain, kadang kita menampilkan variabel dengan mengecek terlebih dahulu apakah variabel tersebut telah dideklarasikan atau belum. Dengan blade template, kita dapat memodifikasi sintak yang ada pada file **index.blade.php** yang berada pada folder **websaya/resources/views/mhs/** menjadi seperti berikut:

```
<!DOCTYPE html>
<html>
    <head>
        <title>WebSaya.Com</title>
    </head>
    <body>
        <h1>Selamat Datang {{ isset($mhs) ? $mhs : 'Tidak ada'  }} di WebSaya.Com</h1>
    </body>
</html>
```

Dan jika dijalankan kembali hasilnya adalah seperti berikut:


Pada hasil diatas memang tidak terdapat perbedaan dari sebelumnya, namun sebernarnya dengan cara tersebut, jika variabel mhs telah dideklarasikan, maka akan ditampilkan nilai variabel mhs, sedangkan jika belum dideklarasikan maka akan menampilkan text default "Tidak ada".

## CONTROL STRUKTUR PADA BLADE TEMPLATE

Blade template juga menyediakan cara tersendiri untuk melakukan logika percabangan maupun perulangan. Untuk melakukan percabangan dengan blade template dapat menggunakan **@if.. @else.. @endif.** Untuk mencobanya buatlah satu route baru pada file **web.php** yang berada pada folder **routes/web.php** dengan sintak sebagai berikut:

```
Route::get('/mhs/show', [mhsController::class, 'show'])->name('mhs-show');
```

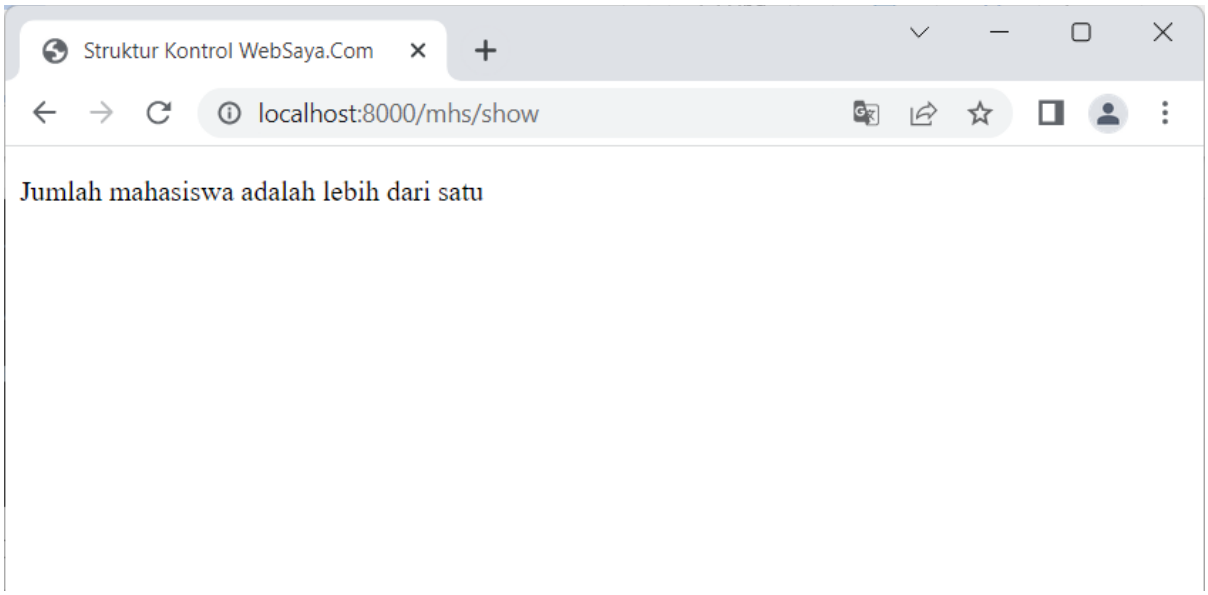
Kemudian pada class controller **mhsController.php** yang berada pada folder **app\Http\Controller** tambahkan satu fungsi baru bernama **show()** dan isikan script sebagai berikut:

```php
public function show(){
    $mhs = ['Adeeva','Nadia','Zahra'];
    return view('mhs/show',compact('mhs'));
}
```

Selanjutnya buatlah view baru dengan nama **show.blade.php** pada folder **resources/views/mhs** dan isikan skript seperti dibawah:

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Struktur Kontrol WebSaya.Com</title>
    </head>
    <body>
        @if (count($mhs) == 1)
            <p>Jumlah mahasiswa adalah satu</p>
        @elseif (count($mhs) == 2)
            <p>Jumlah mahasiswa adalah dua</p>
        @else
            <p>Jumlah mahasiswa adalah lebih dari satu</p>
        @endif
    </body>
</html>
```

Jika dicoba dijalankan dengan mengakses link http://**localhost:8000/mhs/show** maka tampilannya akan menjadi seperti pada gambar dibawah:



Sedangkan untuk perulangan pada blade template mendukung berbagai jenis perulangan, dengan contoh sebagai berikut:

Perulangan dengan @for, @foreach dan @while, untuk mempraktekan penggunaan perulangan menggunakan blade template kita masih akan menggunakan route http://**localhost:8000/mhs/show**. Pada class Controller **mhsController.php** di dalam method show() kita telah mendifisikan satu array bernama mhs. Buat sebuah route baru seperti dibawah ini:

```
Route::get('/mhs/perulangan', [mhsController::class,
        'perulangan'])->name('mhs-perulangan');
```

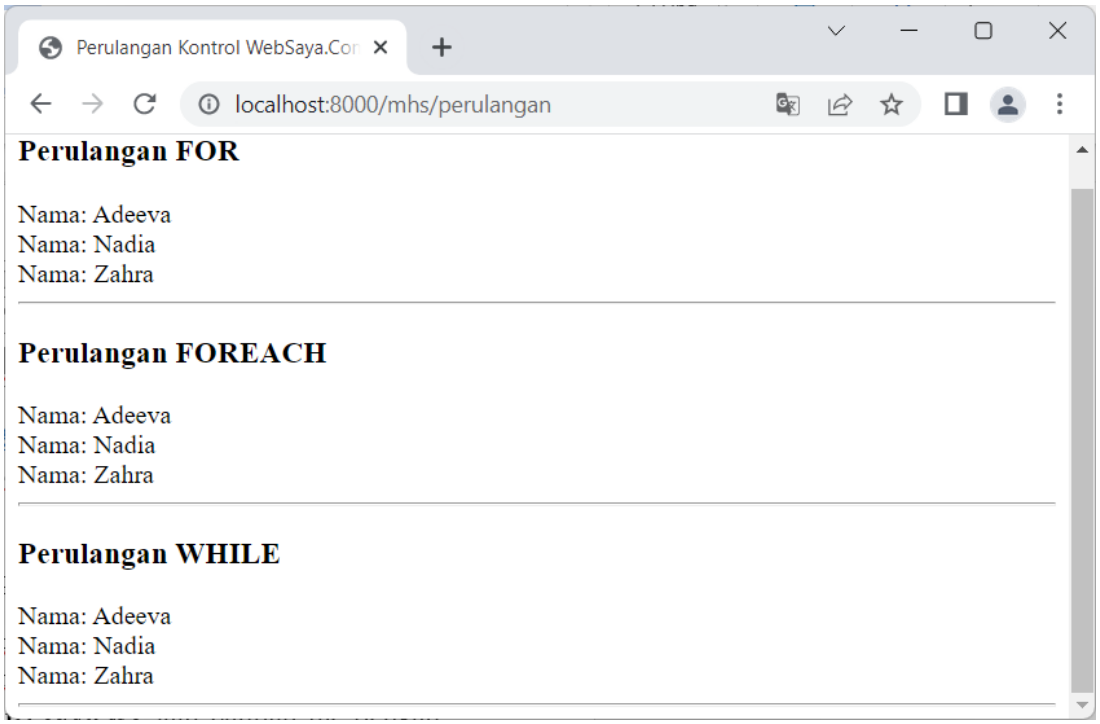
Tambahkan fungsi baru di mhsController seperti berikut:

```
public function perulangan(){
    $mhs = ['Adeeva','Nadia','Zahra'];
    return view('mhs/perulangan', compact('mhs'));
}
```

Buat halaman view/mhs perulangan dengan nama **perulangan.blade.php** seperti berikut:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Perulangan Kontrol WebSaya.Com</title>
    </head>
    <body>
        <h3>Perulangan FOR</h3>
        @for($i=0; $i < count($mhs); $i++)
            {{ 'Nama: '. $mhs[$i] }} <br>
        @endfor
        <hr>
        <h3>Perulangan FOREACH</h3>
        @foreach ($mhs as $item)
            {{ 'Nama: '. $item }} <br>
        @endforeach
        <hr>
        <h3>Perulangan WHILE</h3>
        @php
            $x=0;
        @endphp
        @while($x < count($mhs))
            {{ 'Nama: '. $mhs[$x] }} <br>
            @php
            $x++;
            @endphp
        @endwhile
        <hr>
    </body>
</html>
```

Jika halaman pada link http://**localhost:8000/mhs/perulangan** di refresh atau dimuat ulang maka halamanya akan nampak seperti pada gambar dibawah;




## LAYOUT DENGAN BLADE TEMPLATE

Untuk membuat layout pada template aplikasi, Laravel menggunakan cara yang mudah menggunakan blade template. Untuk mempraktekannya buatlah satu **folder baru** yang bernama **layouts** pada folder **resourcess/views**, lalu buatlah file dengan nama **main.blade.php** menggunakan framework bootsrap disini di dalamnya sesuaikan script seperti berikut ini:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@yield('title')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/boot
  </head>
  <body>
    <header>
        @include('layouts.header')
    </header>
    @include('layouts.nav')
    <div class="container">
        @yield('content')
    </div>
    <footer>
        @include('layouts.footer')
    </footer>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/boot
  </body>
</html>
```

Keterangan dari masing-masing script diatas adalah sebagai berikut:

1. @yield('title') diartikan kita menyediakan tempat yang datanya akan diisi oleh file lain menggunakan @section('title'). Begitu juga dengan script yield('content') akan diisi oleh file blade lain dengan @section('content'). Data data bagian ini bersifat dinamis sesuai dengan halaman yang sedang dibuka.
2. @inlcude('layouts.header') diartikan kita memanggil file header.blade.php pada folder layouts untuk ditampilkan pada halaman tersebut.
3. @include('layouts.nav') diartikan kita memanggil file nav.blade.php

Selanjutnya buatlah satu file baru pada folder yang sama yaitu resources/views/layouts dengan nama header.blade.php dan isikan script sebagai berikut:

```
<div class="alert alert-primary mb-0" role="alert">
    <div class="container">
        <h1>WebSaya.Com</h1>
    </div>
</div>
```

Selanjutnya buatlah satu file baru pada folder yang sama yaitu resources/views/layouts dengan nama **nav.blade.php** dan isikan script sebagai berikut:

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container">
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarNav" aria-controls="navbarNav"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link {{ ($slug === "home") ? 'active' : '' }}"
                aria-current="page" href="/home">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link {{ ($slug === "profil") ? 'active' : '' }}"
                href="/profil">Profil</a>
          </li>
          <li class="nav-item">
            <a class="nav-link {{ ($slug === "mahasiswa") ? 'active' : '' }}"
                href="/mahasiswa">Mahasiswa</a>
          </li>
          <li class="nav-item">
            <a class="nav-link {{ ($slug === "prodi") ? 'active' : '' }}"
                href="/prodi">Prodi</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```

Selanjutnya buatlah satu file baru pada folder yang sama yaitu resources/views/layouts dengan nama **footer.blade.php** dan isikan script sebagai berikut:

```
<div class="container" style="text-align: center">
    Copyright 2022 - websaya.com
</div>
```

Selanjutnya untuk membuat contoh halaman yang menggunakan layout. Buatlah satu folder baru dan berinama **konten** pada folder **resourcess/views** untuk menyimpan

contoh halaman yang akan kita buat. Misalnya kita buat satu file dengan nama **home.blade.php** dengan script sebagai berikut:

Keterangan dari masing-masing script diatas adalah sebagai berikut:

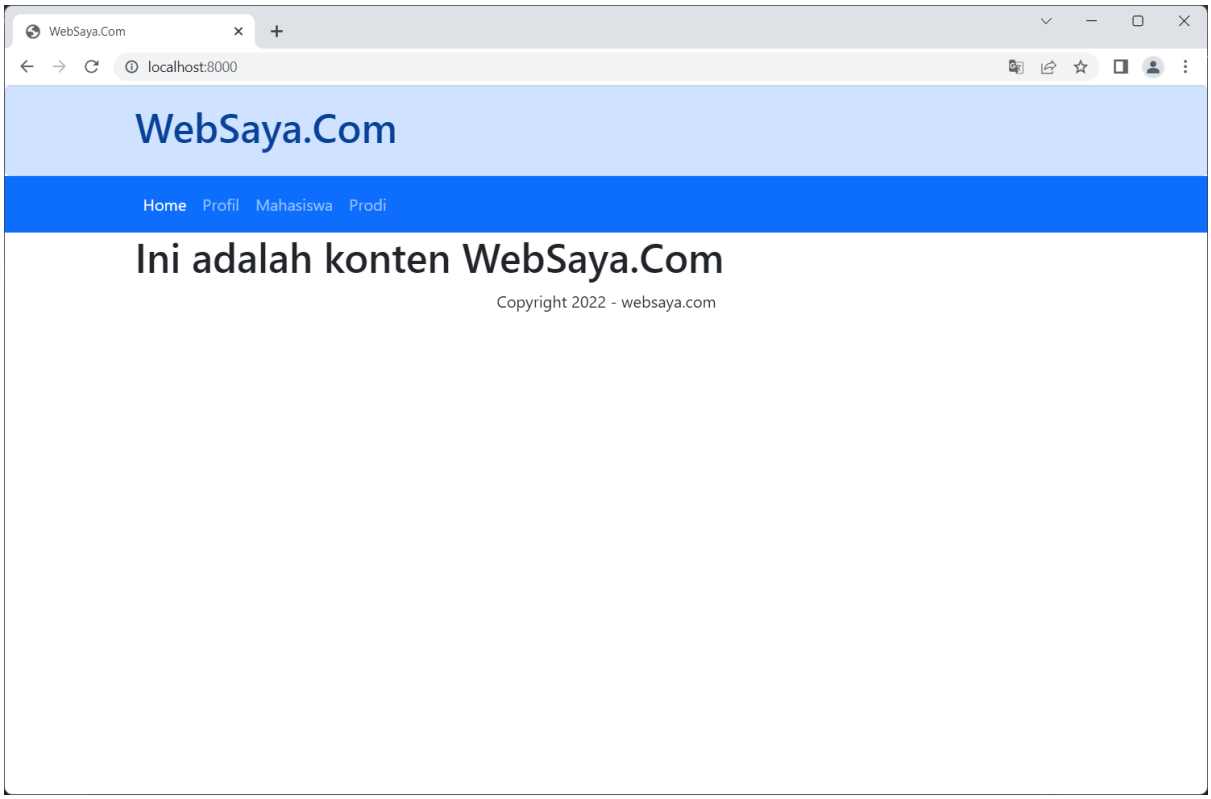
```
@extends('layouts.main')
@section('title',$title)
@section('content')
        <h1>{{ $konten }}</h1>
@endsection
```

1. **@extends('layouts.main)** berarti kita mewarisi semua yang ada pada file main.blade.php pada folder layouts sebagai file master template.
2. **@section('title',$title)** berarti kita menyiapkan data untuk ditampilkan di bagian **yield('title')** pada file master yaitu **main.blade.php.** data **$title** dikirim dari controller atau route yang memaminggil view dengan fungsi view.
3. **@section('content')** ..... **@endsection** diartikan kita menyiapkan data untuk ditampilkan di bagian **@yield('content')** pada file master. data $konten dikirim dari controller atau route yang memaminggil view dengan fungsi view.

Untuk membuktikan hasil dari pembuatan layout diatas, silahkan buat route pada file web.php pada file routes/web.php dengan script sebagai berikut:

```
Route::get('/', function () {
    $title = "WebSaya.Com";
    $slug = "home";
    $konten = "Ini adalah konten WebSaya.Com";
    return view('konten.home', compact('title','slug','konten'));
});
Route::get('/home', function(){
    $title = "WebSaya.Com";
    $slug = "home";
    $konten = "Ini adalah konten WebSaya.Com";
    return view('konten.home', compact('title','slug','konten'));
});
```

Pada script diatas kita membuat variabel $title, $slug dan $konten yang akan dikirimkan ke view dengan fungsi compact(). Jika kita membuka http://**localhost:8000/home** maka hasilnya akan nampak seperti gambar dibawah:




## TUGAS

1. Buatlah 3 halaman yang saling terhubung satu dengan yang lainnya view dan blade
    - Profil: Menampikkan profil mahasiswa yang melakukan praktik
    - Mahasiswa: Tabel data mahasiswa minimal 5 baris
    - Prodi: Menampilkan data semua program studi POLINDRA
2. Buatlah layout halamannya dengan menggunakan master template blade

Catatan:

- Langkah-langkah pembuatannya dapat mengikuti praktikum
- Laporan praktikum tugas dan sourcode dikumpulkan ke elearning