

**LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT**


Pengantar dan Instalasi Laravel



**Oleh:
MAHENDRA FIRMANSYAH
NIM. 1841720027
TI-2A**

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG**

Praktikum – Bagian 1: Instalasi Laravel di Windows

Langkah	Keterangan
1	<p>Laravel menggunakan Composer untuk mengelola dependensinya. Jadi, sebelum menggunakan Laravel, pastikan Composer telah terinstall terlebih dahulu. Composer adalah dependensi manajer untuk library PHP. Composer digunakan untuk menginstal, menghapus, dan memperbarui paket PHP. Unduh composer melalui https://getcomposer.org/, pilih Download untuk memulai proses unduh.</p> 
2	<p>Terdapat 2 cara dalam menginstall Laravel,</p> <p>1. Install Via Laravel Installer</p> <p>Setelah proses instalasi Composer selesai, buka Command Prompt lalu tujulah lokasi folder yang akan menampung folder kerja kita. Karena pada praktikum sebelumnya telah</p>

menggunakan XAMPP, maka dapat menggunakan folder htdocs. Pindah ke direktori htdocs dengan ketik: **cd C:\xampp\htdocs**. Setelah berada di dalam folder, ketikkan perintah sebagai berikut:

composer global require "laravel/installer"

```
C:\xampp\htdocs>composer global require "laravel/installer"
Changed current directory to C:/Users/Mahendra/AppData/Roaming/Composer
Using version ^3.0 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 14 installs, 0 updates, 0 removals
- Installing symfony/process (v5.0.7): Downloading (100%)
- Installing symfony/polyfill-ctype (v1.15.0): Downloading (100%)
- Installing symfony/filesystem (v5.0.7): Downloading (100%)
- Installing psr/container (1.0.0): Downloading (100%)
- Installing symfony/service-contracts (v2.0.1): Downloading (100%)
- Installing symfony/polyfill-php73 (v1.15.0): Downloading (100%)
- Installing symfony/polyfill-mbstring (v1.15.0): Downloading (100%)
- Installing symfony/console (v5.0.7): Downloading (100%)
- Installing ralouphie/getallheaders (3.0.3): Downloading (100%)
- Installing psr/http-message (1.0.1): Downloading (100%)
- Installing guzzlehttp/psr7 (1.6.1): Downloading (100%)
- Installing guzzlehttp/promises (v1.3.1): Downloading (100%)
- Installing guzzlehttp/guzzle (6.5.2): Downloading (100%)
- Installing laravel/installer (v3.0.1): Downloading (100%)
symfony/service-contracts suggests installing symfony/service-implementation
symfony/console suggests installing symfony/event-dispatcher
symfony/console suggests installing symfony/lock
symfony/console suggests installing psr/log (For using the console logger)
guzzlehttp/psr7 suggests installing zendframework/zend-httpdierrunner (Emit PSR-7 responses)
guzzlehttp/guzzle suggests installing psr/log (Required for using the Log middleware)
guzzlehttp/guzzle suggests installing ext-intl (Required for Internationalized Domain Name (IDN) support)
Writing lock file
Generating autoload files
6 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

Tunggulah hingga proses penginstalan selesai. Setelah itu, Kita bisa mulai membuat project laravel dengan mengetikkan perintah:

laravel new laravelapp

```
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
> @php artisan key:generate --ansi
Application key set successfully.
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
Application ready! Build something amazing.
```

Keterangan:

laravelapp: merupakan folder kerja laravel, Kita dapat menggantinya sesuai dengan keinginan

2. Install Via Composer Create-Project

Selain menggunakan cara pertama, Kita dapat melakukan penginstalan laravel dengan mengetikkan perintah berikut pada command prompt:

composer create-project --prefer-dist laravel/laravel laravelapp

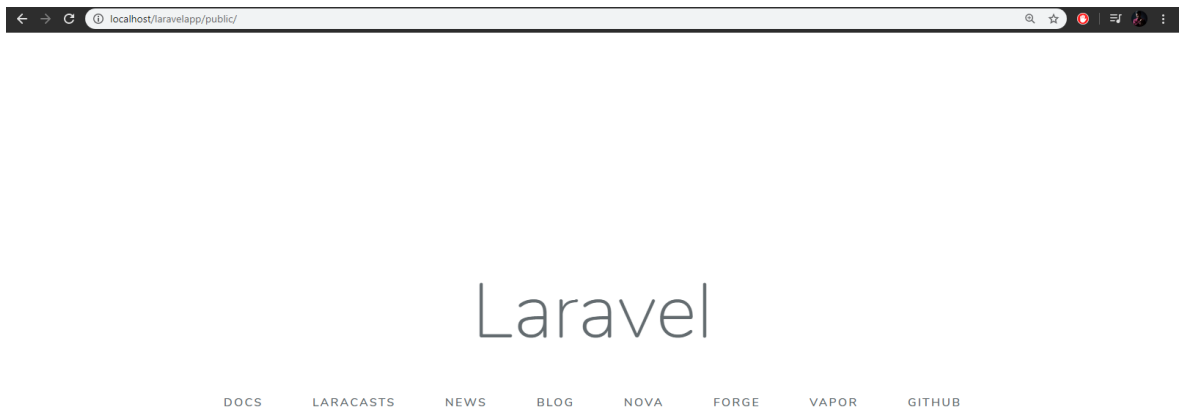
3

Setelah proses instalasi Laravel selesai, Kita perlu menguji apakah hasil instalasi tersebut berjalan dengan baik atau tidak. Aktifkan Apache server lewat XAMPP Control Panel. Pada browser, ketik alamat <http://localhost/laravelapp/public/> Atau Kita dapat menggunakan perintah berikut di Command Prompt:

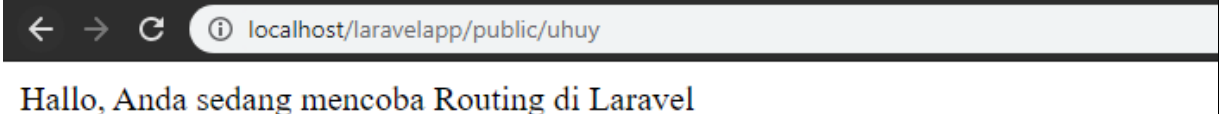
php artisan serve

```
PS C:\xampp\htdocs> cd .\laravelapp\  
PS C:\xampp\htdocs\laravelapp> php .\artisan serve  
Laravel development server started: <http://127.0.0.1:8000>
```

Kita dapat menjalankan aplikasi LARAVEL di folder manapun dengan menggunakan perintah di atas sehingga tidak harus selalu berada di dalam folder htdocs (XAMPP). Jalankan dengan localhost:8000, dan pastikan tampilan di browser akan seperti berikut ini :



Praktikum – Bagian 2: Mengenal dan membuat route

Langkah	Keterangan
1	<p>Setiap request yang datang pada laravel akan diarahkan melalui sebuah route. Route ini yang akan menentukan respon apa yang akan dikerjakan untuk membalas request tersebut. Halaman homepage default Laravel seperti yang ditampilkan pada gambar di Bagian 1 adalah hasil dari script yang terletak pada file laravelapp\routes\web.php Berikut ini adalah perintah route untuk menampilkan halaman default Laravel tersebut</p> <pre>1 <?php 2 3 /* 4 ----- 5 Web Routes 6 ----- 7 8 Here is where you can register web routes for your application. These 9 routes are loaded by the RouteServiceProvider within a group which 10 contains the "web" middleware group. Now create something great! 11 12 */ 13 14 Route::get('/', function () { 15 return view('welcome'); 16 }); 17</pre>
2	<p>Kita dapat menuliskan perintah baru untuk membuat route, selain mengembalikan file view sebagai response, Kita dapat mengembalikan response berupa string atau teks biasa. Buka file laravelapp\routes\web.php, tambahkan isinya dengan script di bawah ini :</p> <pre>Route::get('/uhuy', function () { return "Hallo, Anda sedang mencoba Routing di Laravel"; });</pre> <p>Untuk melihat hasilnya, pada browser, ketikkan alamat http://localhost/laravelapp/public/hallo</p> 

3

Selanjutnya, kita akan mencoba membuat route baru dengan menampilkan halaman view. Tambahkan route baru untuk halaman “profil” dengan menambahkan script berikut:

```
22 Route::get('/profil', function () {  
23     return view('profil');  
24 });  
25
```

Buat file baru bernama profil.blade.php di dalam folder laravelapp\resources\views\, ketikkan script berikut ini:

```
web.php  profil.blade.php X  
resources > views > profil.blade.php  
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4     <title> Profil Saya </title>  
5 </head>  
6 <body>  
7     <h1>Profil Saya</h1>  
8     <p> Perkenalkan Nama saya Mahendra, saya seorang mahasiswa</p>  
9 </body>  
10 </html>
```

Tuliskan profil kalian masing-masing. Untuk menampilkan hasilnya, ketikkan alamat <http://localhost/laravelapp/public/profil>

localhost/laravelapp/public/profil

Profil Saya

Perkenalkan Nama saya Mahendra, saya seorang mahasiswa

Praktikum – Bagian 3: Cara Menggunakan Controller pada Framework Laravel

Langkah	Keterangan
1	<p>Pada pembahasan sebelumnya, Kita sudah bisa menampilkan file view langsung dari Routenya. Tetapi sebuah aplikasi tidaklah sesimpel itu, akan ada logika dan data-data yang harus diolah terlebih dahulu. Caranya adalah dengan menggunakan Controller, seperti yang telah dipelajari ketika menggunakan CodeIgniter.</p> <p>Ada dua cara yang dapat digunakan untuk membuat Controller. Cara pertama, dengan membuat file controller secara manual dan tuliskan code untuk extends controller secara manual. Cara kedua, adalah dengan membuat file Controller lewat Artisan di Laravel.</p> <p>Kita gunakan cara yang kedua, pada latihan kali ini kita akan membuat Controller dengan nama CobaController dengan menuliskan di command prompt / terminal :</p> <p style="text-align: center;">php artisan make:controller CobaController</p> <pre>C:\xampp\htdocs\laravelapp>php artisan make:controller CobaController Controller created successfully. C:\xampp\htdocs\laravelapp></pre> <p>Maka akan terbentuk sebuah file dengan nama CobaController.php pada folder laravelapp\app\Http\Controllers\</p> <pre>1 <?php 2 3 namespace App\Http\Controllers; 4 5 use Illuminate\Http\Request; 6 7 class CobaController extends Controller 8 { 9 // 10 } 11</pre>
2	<p>Ubah route halaman 'profil' pada praktikum sebelumnya menjadi seperti berikut</p> <p style="text-align: center;">Route::get('profil', 'CobaController@profil');</p> <pre>22 Route::get('profil', 'CobaController@profil');</pre> <p>Langkah selanjutnya, tambahkan fungsi / method profil() pada class CobaController seperti berikut ini:</p> <pre><?php namespace App\Http\Controllers; use Illuminate\Http\Request; class CobaController extends Controller { public function profil() {</pre>

```
        return view('profil');  
    }  
}
```

```
1  <?php  
2  
3  namespace App\Http\Controllers;  
4  
5  use Illuminate\Http\Request;  
6  
7  class CobaController extends Controller  
8  {  
9      public function profil()  
10     {  
11         return view('profil');  
12     }  
13 }  
14
```

Ketikkan alamat <http://localhost/laravelapp/public/profil>, dan Kita akan mendapatkan hasil yang sama seperti pada latihan sebelumnya.

← → ↻ ⓘ localhost/laravelapp/public/profil

Profil Saya

Perkenalkan Nama saya Mahendra, saya seorang mahasiswa

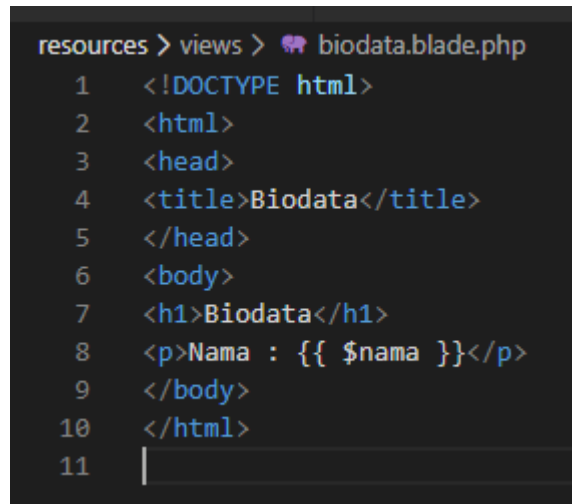
Praktikum – Bagian 4: Memberikan Data Controller kepada View

Langkah	Keterangan
1	<p>Dalam sebuah aplikasi, sangat jarang Kita temukan isi dari file view yang bersifat statis. Pada umumnya, view dipakai untuk menampilkan data, contohnya adalah data yang berasal dari database. Namun, karena kita belum sampai pada pembahasan database, maka pada latihan kali ini kita akan mencoba menampilkan data dari variabel.</p> <p>Buat route baru untuk halaman biodata</p> <pre>Route::get('biodata', 'BiodataController@index');</pre> <p>Buat controller baru dengan nama BiodataController.php</p> <pre>php artisan make:controller BiodataController</pre> <p>Ketikkan script berikut dengan memberikan method / fungsi index pada laravelapp\app\Http\Controllers\BiodataController.php</p> <pre><?php namespace App\Http\Controllers; use Illuminate\Http\Request; class BiodataController extends Controller { public function index() { \$nama = 'Sugono Galih Aprianto'; //ubah dengan nama kalian return view('biodata' , ['nama' => \$nama]); } }</pre> 

2

Buat file view dengan nama **biodata.blade.php** pada **laravelapp\resources\views**

```
<!DOCTYPE html>
<html>
<head>
    <title>Biodata</title>
</head>
<body>
    <h1>Biodata</h1>
    <p>Nama : {{ $nama }}</p>
</body>
</html>
```

A screenshot of a code editor with a dark background. The file path 'resources > views > biodata.blade.php' is shown at the top. The code content is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Biodata</title>
5 </head>
6 <body>
7 <h1>Biodata</h1>
8 <p>Nama : {{ $nama }}</p>
9 </body>
10 </html>
11 |
```

Untuk melihat hasilnya, pada browser, ketik alamat <http://localhost:8000/biodata> .



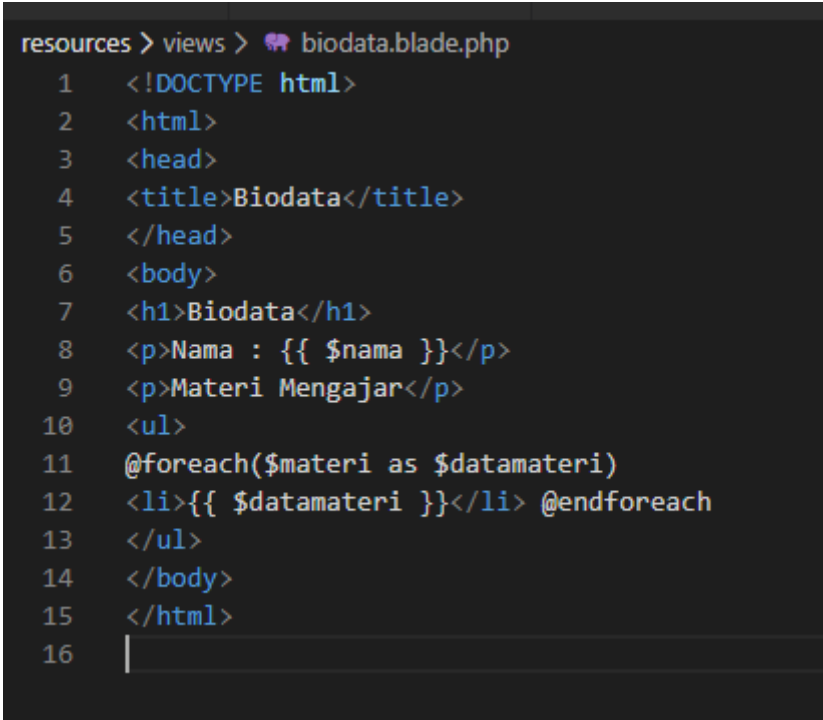
Praktikum – Bagian 5: Memberikan Data Array Kepada View

Langkah	Keterangan
1	<p>Pada bagian ini, Kita coba untuk memberikan data array kepada view. Tambahkan variabel data array pada BiodataController.php</p> <pre> <?php namespace App\Http\Controllers; use Illuminate\Http\Request; class BiodataController extends Controller { public function index() { \$nama = 'Sugono Galih Aprianto'; <i>//ubah dengan nama kalian</i> \$materi = ["Web Design", "Web Programming", "Digital Marketing","Graphic Design"]; return view('biodata' , ['nama' => \$nama, 'materi' => \$materi]); } } </pre> <p>The screenshot shows a code editor with the following content:</p> <pre> app > Http > Controllers > BiodataController.php 1 <?php 2 namespace App\Http\Controllers; 3 use Illuminate\Http\Request; 4 class BiodataController extends Controller 5 { 6 public function index() { 7 \$nama = 'Mahendra Firmansyah'; <i>//ubah dengan nama kalian</i> 8 \$materi = ["Web Design", "Web Programming", "Digital 9 Marketing","Graphic Design"]; 10 return view('biodata' , ['nama' => \$nama, 'materi' => 11 \$materi]); 12 } 13 } 14 </pre>

2

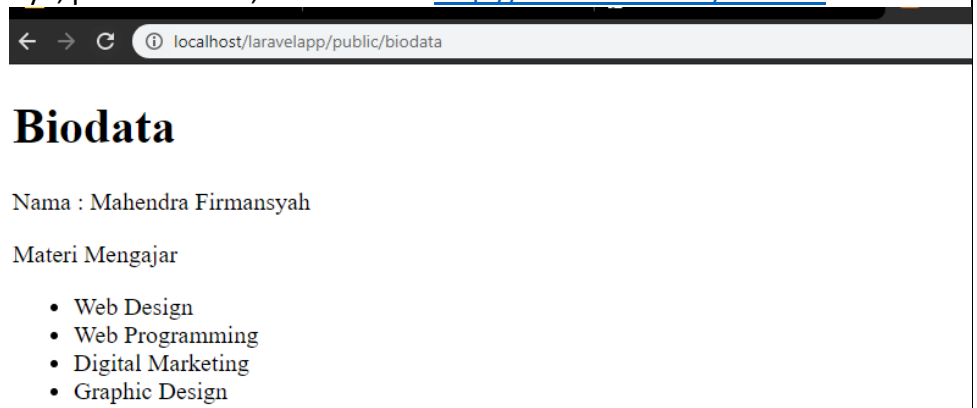
Tambahkan isi file view **biodata.blade.php** menjadi seperti berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>Biodata</title>
</head>
<body>
    <h1>Biodata</h1>
    <p>Nama : {{ $nama }}</p>
    <p>Materi Mengajar</p>
    <ul>
        @foreach($materi as $datamateri)
            <li>{{ $datamateri }}</li>
        @endforeach
    </ul>
</body>
</html>
```

A screenshot of a code editor with a dark background. The file path 'resources > views > biodata.blade.php' is visible at the top. The code is a Blade template for a biodata page, containing HTML tags for a head section with a title 'Biodata' and a body section with an h1 'Biodata', two paragraphs for 'Nama' and 'Materi Mengajar', and a list of 'materi' items. The code is numbered from 1 to 16 on the left side of the editor.

```
resources > views > biodata.blade.php
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Biodata</title>
5  </head>
6  <body>
7  <h1>Biodata</h1>
8  <p>Nama : {{ $nama }}</p>
9  <p>Materi Mengajar</p>
10 <ul>
11 @foreach($materi as $datamateri)
12 <li>{{ $datamateri }}</li> @endforeach
13 </ul>
14 </body>
15 </html>
16 |
```

Untuk melihat hasilnya, pada browser, ketik alamat <http://localhost:8000/biodata> .



-- Selamat Mengerjakan --