

Nama : Akmal Muhamad Firdaus

NIM : 1301204188

Praktikum ABP Modul 6

Pendahuluan

1. Framework & MVC

Framework adalah kumpulan dari aturan, standar, dan konvensi yang membantu dalam mengembangkan aplikasi web dengan lebih mudah dan efisien. Framework menyediakan kerangka kerja yang telah disediakan sebelumnya untuk memudahkan pengembangan aplikasi, sehingga pengembang dapat fokus pada masalah bisnis yang spesifik.

MVC (Model-View-Controller) adalah pola arsitektur yang sering digunakan dalam pengembangan aplikasi web. MVC memecah aplikasi menjadi tiga komponen utama:

- a. Model - Merupakan bagian dari aplikasi yang bertanggung jawab untuk mengelola data dan aturan bisnis aplikasi.
- b. View - Merupakan bagian dari aplikasi yang bertanggung jawab untuk menampilkan data kepada pengguna akhir melalui antarmuka pengguna.
- c. Controller - Merupakan bagian dari aplikasi yang bertanggung jawab untuk menangani permintaan pengguna dan menghubungkan Model dan View.

Dengan menggunakan pola arsitektur MVC, pengembang dapat memisahkan tugas-tugas dalam pengembangan aplikasi web dan meningkatkan kemampuan aplikasi untuk diuji dan dikelola dengan lebih efisien. Banyak framework PHP yang mendukung pola arsitektur MVC, contohnya Laravel.

2. Pengenalan Laravel

Laravel adalah sebuah framework PHP yang open source dan digunakan untuk membangun aplikasi web. Framework ini dikembangkan oleh Taylor Otwell pada tahun 2011 dan telah menjadi salah satu framework PHP yang paling populer digunakan oleh pengembang web.

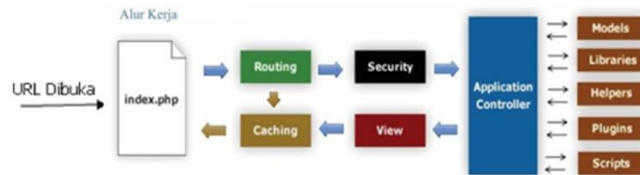
Laravel didesain untuk memudahkan pengembangan aplikasi web dengan menggunakan pola arsitektur Model-View-Controller (MVC). Framework ini menyediakan banyak fitur yang membantu pengembang untuk membangun aplikasi web yang kompleks dengan cepat dan efisien, termasuk fitur-fitur seperti routing, migrasi database, ORM, templating engine, dan banyak lagi.

Beberapa kelebihan dari Laravel antara lain:

- Mudah dipelajari dan digunakan oleh pengembang web pemula maupun berpengalaman.
- Free, karena berada di bawah lisensi open source, kita dapat melakukan apa pun
- Memiliki dokumentasi yang lengkap dan dukungan komunitas yang besar.
- Memiliki sistem routing yang mudah digunakan dan dapat disesuaikan dengan kebutuhan aplikasi.
- Memiliki fitur bawaan yang kuat untuk otentikasi dan otorisasi pengguna.

- Memiliki sistem templating yang kuat untuk membangun tampilan aplikasi web.
- Mendukung banyak database management system (DBMS) seperti MySQL, PostgreSQL, dan MongoDB.

3. Cara Kerja Laravel



Soruce: Modul Praktikum ABP

Ketika suatu URL diakses, Laravel akan membaca file index.php dan memeriksa apakah URL tersebut terdaftar pada routing yang sudah dibuat. Jika URL tersebut terdaftar pada routing dan memenuhi persyaratan keamanan (middleware), seperti autentikasi **jika diperlukan**, maka Laravel akan memanggil controller yang sesuai dengan routing tersebut. Di dalam controller, semua logika bisnis yang diperlukan akan dieksekusi dan dapat mengembalikan view ke browser. Tampilan tersebut akan disimpan dalam cache agar pemrosesan selanjutnya lebih cepat.

Laravel memiliki banyak fitur bawaan yang memudahkan pengembang untuk membangun aplikasi web dengan lebih efisien, seperti sistem autentikasi dan otorisasi, migrasi database, dan masih banyak lagi.

Selain itu, Laravel juga memiliki fitur composer, yang merupakan manajer paket pada PHP. Fitur ini memudahkan pengembang untuk mengelola dependensi pada aplikasi web, seperti memasang library dan package pihak ketiga.

Secara keseluruhan, Laravel bekerja dengan menggabungkan pola arsitektur MVC, routing, dan fitur-fitur bawaan yang efisien dan mudah digunakan oleh pengembang web PHP.

```

> app
> bootstrap
> config
> database
> lang
> public
> resources
> routes
> storage
> tests
> vendor
  
```

Struktur Folder Laravel

1. app/

Folder ini berisi kode aplikasi Laravel, seperti model, controller, middleware, dan provider.

2. bootstrap/

Folder ini berisi file yang digunakan untuk memulai aplikasi Laravel, seperti file app.php dan autoload.php.

3. config/

Folder ini berisi konfigurasi untuk aplikasi Laravel, seperti konfigurasi database, cache, session, dan lain-lain.

4. database/

Folder ini berisi migrasi dan seed untuk database aplikasi Laravel. Migrasi digunakan untuk membuat struktur tabel database, sedangkan seed digunakan untuk memasukkan data awal ke dalam database.

5. lang/

Laravel adalah framework yang support localization (multi bahasa), untuk menerapkan berbagai bahasa, dapat disimpan pada folder lang ini.

6. public/

Folder ini berisi file yang dapat diakses secara publik melalui browser, seperti file index.php dan file gambar.

7. resources/

Folder ini berisi aset-aset yang digunakan dalam aplikasi Laravel, seperti file tampilan (view), file bahasa, dan file konfigurasi lainnya.

8. storage/

Folder ini berisi file sementara dan file yang digunakan oleh Laravel, seperti file log dan file cache.

9. tests/

Folder ini berisi file untuk melakukan pengujian aplikasi Laravel.

10. vendor/

Folder ini berisi paket-paket pihak ketiga yang digunakan oleh aplikasi Laravel.

Secara keseluruhan, struktur folder Laravel dirancang agar mudah dipahami dan diorganisasi. Setiap folder memiliki tugas dan fungsinya masing-masing sehingga memudahkan pengembangan aplikasi Laravel.


4. Instalasi Laravel

Pastikan terlebih dahulu, bahwa [composer](#) sudah terinstall. Lalu jalankan perintah “composer global require laravel/installer” untuk menginstall laravel secara global.

```
C:\xampp\htdocs\lab8>composer global require laravel/installer
Using version ^2.0 for laravel/installer
./composer.json has been created
Running composer require laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 18 installs, 0 updates, 0 removals
  - Locking laravel/installer (v2.0.0)
  - Locking psr/container (2.0.2)
  - Locking symfony/console (v6.4.10)
  - Locking symfony/http-foundation (v6.4.10)
  - Locking symfony/http-kernel (v6.4.10)
  - Locking symfony/mime (v6.4.10)
  - Locking symfony/process (v6.4.10)
  - Locking symfony/string (v6.4.10)
  - Locking symfony/yaml (v6.4.10)
Installing dependencies from lock file (including require-dev)
Package operations: 18 installs, 0 updates, 0 removals
  - Downloading symfony/process (v6.4.10): Extracting archive
  - Downloading symfony/http-foundation (v6.4.10): Extracting archive
  - Downloading symfony/http-kernel (v6.4.10): Extracting archive
  - Downloading symfony/mime (v6.4.10): Extracting archive
  - Downloading symfony/string (v6.4.10): Extracting archive
  - Downloading symfony/yaml (v6.4.10): Extracting archive
  - Downloading symfony/console (v6.4.10): Extracting archive
  - Downloading psr/container (2.0.2): Extracting archive
  - Downloading laravel/installer (v2.0.0): Extracting archive
  - Installing symfony/process (v6.4.10): Extracting archive
  - Installing symfony/http-foundation (v6.4.10): Extracting archive
  - Installing symfony/http-kernel (v6.4.10): Extracting archive
  - Installing symfony/mime (v6.4.10): Extracting archive
  - Installing symfony/string (v6.4.10): Extracting archive
  - Installing symfony/yaml (v6.4.10): Extracting archive
  - Installing symfony/console (v6.4.10): Extracting archive
  - Installing psr/container (2.0.2): Extracting archive
  - Installing laravel/installer (v2.0.0): Extracting archive
A package is not in the composer.recommended section. You may want to check the package's repository to see details.
No security advisories found for the installed packages.
```

Jalankan perintah `laravel new "nama_project"`, lalu tunggu hingga proses pembuatan proyek baru selesai.

```
> composer create-project laravel/laravel new "Model v"
```



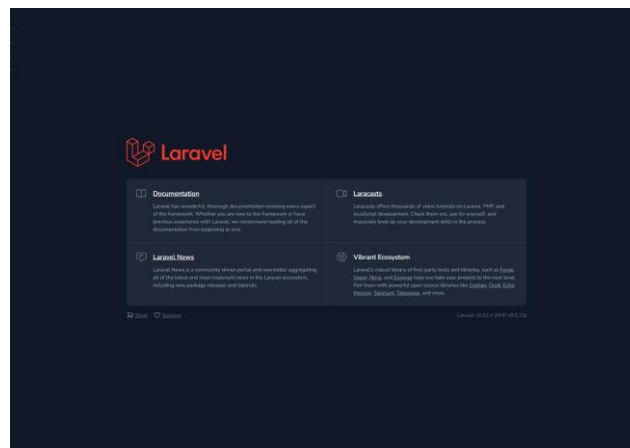
Creating a "laravel/laravel" project at "/. /Model v"
Cannot use laravel/laravel's latest version v10.0.0 as it requires php >=8.1 which is not satisfied by your platform.

```
Installing laravel/laravel (v9.5.2)
-> Downloading laravel/laravel (v9.5.2)
-> Installing laravel/laravel (v9.5.2) [Extracting archive]
Created project in C:\xampp\htdocs\ASP\Model v
> composer require "C:\xampp\htdocs\ASP\Model v"
> composer req "file-exists-@env" || copy("env.example", "env");"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 136 installs, 0 updates, 0 removals
  - Locking brick/math (0.10.2)
  - Locking dflydev/dot-access-data (v3.0.2)
  - Locking doctrine/instantiator (2.0.9)
  - Locking doctrine/reflection (2.0.6)
  - Locking doctrine/sql-formatter (1.5.0)
  - Locking dragonmantank/cron-expression (v3.2.0)
  - Locking egulias/email-validator (3.2.5)
  - Locking fakerphp/faker (v1.21.0)
  - Locking filp/whoops (2.15.1)
  - Locking fruitcake/php-cors (v1.2.0)
  - Locking GrahamCampbell/result-type (v1.1.1)
  - Locking guzzlehttp/guzzle (7.5.0)
  - Locking guzzlehttp/promises (1.5.2)
  - Locking guzzlehttp/psr7 (2.6.0)
  - Locking guzzlehttp/uri-template (v1.0.1)
  - Locking hamcrest/hamcrest-php (v2.0.1)
  - Locking laravel/framework (v9.52.0)
```

Lalu jalankan “php artisan ser” untuk menjalankan server

```
C:\xampp\htdocs\ABP>cd ..  
C:\xampp\htdocs\ABP>cd "Modul 6"  
C:\xampp\htdocs\ABP\Modul 6>php artisan ser  
  
INFO Server running on [http://127.0.0.1:8000].  
  
Press Ctrl+C to stop the server
```

Welcome page Laravel



5. Model

Pembuatan model menggunakan CLI (Command Line Interface) pada Laravel sangat mudah dilakukan. Pertama, buka terminal atau command prompt dan pastikan sudah berada di dalam folder proyek Laravel. Kemudian, ketik perintah *php artisan make:model NamaModel* dan tekan Enter.

```
C:\xampp\htdocs\ABP\Modul 6>php artisan make:model Product
```

```
INFO Model [C:\xampp\htdocs\ABP\Modul 6\app/Models/Product.php] created successfully.
```

Setelah itu, Laravel akan membuatkan file model baru pada folder "app/Models". Secara otomatis, nama file model tersebut akan mengikuti aturan penamaan konvensi Laravel, yaitu menggunakan format CamelCase dan singular .

Secara default, model yang baru dibuat akan memiliki beberapa konfigurasi awal seperti pengaturan nama tabel, kolom fillable, primary key, timestamp, dan sebagainya. Konfigurasi tersebut dapat dilihat pada file model yang baru dibuat pada bagian \$table, \$fillable, \$primaryKey, \$timestamps, dan sebagainya.

Selain itu, jika model tersebut akan terhubung ke database, maka Laravel akan memerlukan koneksi database yang sudah terkonfigurasi sebelumnya pada file .env di root folder projek. Dalam file .env, pastikan konfigurasi database sudah sesuai dengan server database yang digunakan. Jika belum terkonfigurasi, maka Laravel akan menghasilkan error saat menggunakan model tersebut.

Dikarenakan pada modul ini belum ada penggunaan database pada model, maka model dibuatkan data dummynya menggunakan array menjadi seperti berikut:



```
1 <?php
2 namespace App\Models;
3 use Illuminate\Database\Eloquent\Model;
4 class Product extends Model
5 {
6     //Author: Akmal Muhamad Firdaus - 1301204188
7     protected $table = 'products';
8     public static function getAllProducts()
9     {
10         $products = [
11             [
12                 'id' => 1,
13                 'name' => 'Product 1',
14                 'description' => 'Description 1',
15                 'price' => 1000
16             ],
17             [
18                 'id' => 2,
19                 'name' => 'Product 2',
20                 'description' => 'Description 2',
21                 'price' => 2000
22             ],
23             [
24                 'id' => 3,
25                 'name' => 'Product 3',
26                 'description' => 'Description 3',
27                 'price' => 3000
28             ]
29         ];
30
31         return $products;
32     }
33     public static function addProduct($data)
34     {
35         $products = self::getAllProducts();
36         $lastId = end($products)['id'];
37         $data['id'] = $lastId + 1;
38         $products[] = $data;
39
40         return $products;
41     }
42 }
```

Product Model

Berikut adalah penjelasan mengenai method getAllProducts, getProductById, dan addProduct pada model:

1. getAllProducts():

Method ini berfungsi untuk mendapatkan semua data produk yang ada dalam database. Pada kasus nyata, method ini akan melakukan query `SELECT * FROM products` pada tabel produk dan mengembalikan hasilnya dalam bentuk array atau collection. Dalam implementasinya, method ini dapat menggunakan method `all()` yang disediakan oleh Eloquent ORM untuk mengambil semua data dari tabel terkait. Tetapi dikarenakan pada modul 6 ini belum ada penggunaan database, maka return value dari array ini masih mengembalikan data static dari array pada variable `$products`

2. addProduct(\$data):

Method ini akan menambahkan data produk baru ke dalam array \$products. Parameter \$data berisi informasi produk seperti nama, deskripsi, harga, dan sebagainya. Dalam implementasinya, method ini akan membuat sebuah instance baru dari model Product, mengisi property-nya dengan data yang diberikan, dan menyimpannya ke dalam array \$products.

6. View

View pada Laravel adalah bagian dari arsitektur Model-View-Controller (MVC) yang bertanggung jawab untuk menampilkan data ke user. Dalam Laravel, view diwakili oleh file-file blade template yang berisi kode HTML, PHP, dan bahasa template Laravel yang khusus, seperti directives dan expressions. View bertanggung jawab untuk menampilkan data yang diberikan oleh controller dan juga untuk menangani interaksi user seperti form input, button, dan hyperlink.

Dalam Laravel, view dapat dipecah menjadi beberapa bagian kecil yang disebut sebagai partials. Partial adalah file template yang digunakan kembali dalam beberapa halaman view yang berbeda. Partial dapat mempermudah pengelolaan kode dan meningkatkan kinerja aplikasi karena menghindari duplikasi kode.

Laravel juga menyediakan fitur-fitur yang memudahkan pembuatan view, seperti inheritance dan section. Inheritance memungkinkan developer untuk membuat view induk yang bisa digunakan kembali dalam beberapa view turunan, sedangkan section memungkinkan developer untuk memisahkan bagian-bagian dari view dan menggabungkannya dalam view lain.

```
1 <@extends('layouts.app')
2 <@section('content')
3
4     <!-- Required meta tags -->
5     <meta charset="utf-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7
8     <!-- Bootstrap CSS -->
9     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
10         integrity="sha384-EVSTLt33HsgtYdEvZwp43h2wpvSt16WAjZq33h5263v26E5V0lHp0Gp2bJ27" crossorigin="anonymous" />
11
12     <link href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.1/css/all.min.css" rel="stylesheet" />
13
14     <title>Add New Product</title>
15
16     <div class="container" style="margin-top: 20px;">
17         <div class="row" style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;">
18             <div class="col-12" style="text-align: center; padding: 5px; background-color: #f2f2f2;">
19                 Click to add data
20             </div>
21             <div class="col-12" style="padding: 10px;">
22                 <table class="table" id="users-table">
23                     <thead>
24                         <tr>
25                             <th>Name</th>
26                             <th>Email</th>
27                             <th>Phone</th>
28                             <th>Address</th>
29                         </tr>
30                     </thead>
31                     <tbody>
32                         <tr>
33                             <td>John Doe</td>
34                             <td>john.doe@example.com</td>
35                             <td>08123456789</td>
36                             <td>Jl. Sudirman No. 123</td>
37                         </tr>
38                     </tbody>
39                 </table>
40             </div>
41         </div>
42         <div class="row">
43             <div class="col-12" style="padding: 10px;">
44                 <div class="card" style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;">
45                     <div class="card-header" style="background-color: #f2f2f2; padding: 5px;>
46                         Add New Product
47                     </div>
48                     <div class="card-body" style="padding: 10px;">
49                         <div class="row">
50                             <div class="col-12" style="margin-bottom: 10px;">
51                                 <input type="text" class="form-control" id="name" value="" placeholder="Name" />
52                             </div>
53                             <div class="col-12" style="margin-bottom: 10px;">
54                                 <input type="text" class="form-control" id="description" value="" placeholder="Description" />
55                             </div>
56                             <div class="col-12" style="margin-bottom: 10px;">
57                                 <input type="text" class="form-control" id="price" value="" placeholder="Price" />
58                             </div>
59                             <div class="col-12" style="margin-bottom: 10px;">
60                                 <input type="text" class="form-control" id="address" value="" placeholder="Address" />
61                             </div>
62                             <div class="col-12" style="text-align: center; margin-top: 10px;">
63                                 <button type="button" class="btn btn-primary" id="add-product">Add Product</button>
64                             </div>
65                         </div>
66                     </div>
67                 </div>
68             </div>
69         </div>
70     </div>
71
72     <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha384-vt888333jw99L+v200d434J7LIMfcpkx0kV01uKy6v56wV9vc6wGd6ZlQy6nQ6"
73         crossorigin="anonymous"></script>
74     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
75         integrity="sha384-M6WtQF8hR737FpSb2b84tY6h3473Pb8ctQ6UI3aC9f8sF84f4tQ6bBk10q6q"
76         crossorigin="anonymous"></script>
77
78     </div>
```

7. Controller

Pembuatan controller pada Laravel dapat dilakukan dengan menggunakan command-line interface (CLI) yang disediakan oleh framework ini. Caranya cukup mudah, yaitu dengan menuliskan command `php artisan make:controller NamaController` dapat juga diberi argument tambahan seperti `--resource`. Kelebihan lain dari menggunakan resource ini adalah kita dapat menambah satu Routing saja untuk menangani semua aksi dari satu Controller

```
C:\xampp\htdocs\ABP\Modul 6>php artisan make:controller ProductController
INFO: Controller [C:\xampp\htdocs\ABP\Modul 6\app\Http\Controllers/ProductController.php] created successfully.
```

Setelah controller berhasil dibuat, kita dapat membuka file controller yang telah dibuat (biasanya berada di direktori `App/Http/Controllers`) dan menambahkan method-method yang dibutuhkan. Method-method tersebut biasanya digunakan untuk memproses request dari client dan memberikan response yang sesuai. Pada controller, kita juga dapat melakukan validasi data, melakukan pengolahan data, dan memanggil method-method pada model untuk melakukan query ke database.

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Product;

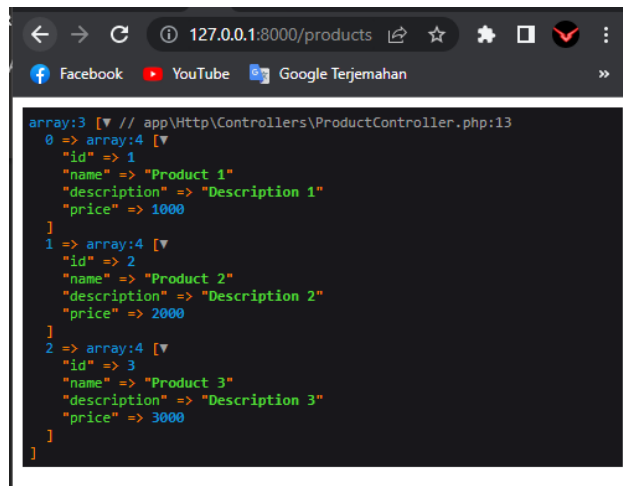
/*
 * @author
 */
class ProductController extends Controller
{
}
```

Setelah controller selesai dibuat, kita dapat menghubungkannya dengan routing pada file `web.php` untuk menentukan URL yang akan dipetakan ke dalam method-method pada controller. Dalam Laravel, kita dapat menggunakan resource routing untuk menghasilkan URL yang sesuai dengan kebutuhan CRUD pada controller.

a. Membuat method index

Method index pada controller product ini digunakan untuk mengambil seluruh data products dari model. Mari kita coba `dd(Dump Die)` untuk melakukan pengecekan bahwa data tersebut sudah diambil dari model.

```
public function index()
{
    $products = Product::getAllProducts();
    dd($products);
    return view('product', compact('products'));
}
```

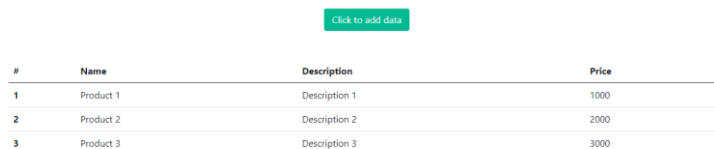


The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/products'. The browser's address bar also shows icons for Facebook, YouTube, and Google Terjemahan. The main content area displays a JSON array of three products, each with an id, name, description, and price. The array is structured as follows:

```
array:3 [▼ // app\Http\Controllers\ProductController.php:13
  0 => array:4 [▼
    "id" => 1
    "name" => "Product 1"
    "description" => "Description 1"
    "price" => 1000
  ]
  1 => array:4 [▼
    "id" => 2
    "name" => "Product 2"
    "description" => "Description 2"
    "price" => 2000
  ]
  2 => array:4 [▼
    "id" => 3
    "name" => "Product 3"
    "description" => "Description 3"
    "price" => 3000
  ]
]
```

Data tersebut sudah sesuai dengan apa yang dikembalikan dari model yaitu array of product. Setelah itu beri comment pada *dd*, supaya nantinya yang ditampilkan adalah view dari halaman product.

Example table products

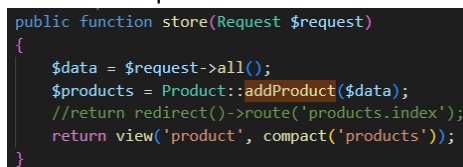


The screenshot shows a web application interface. At the top, there is a green button labeled 'Click to add data'. Below the button is a table with the following data:

#	Name	Description	Price
1	Product 1	Description 1	1000
2	Product 2	Description 2	2000
3	Product 3	Description 3	3000

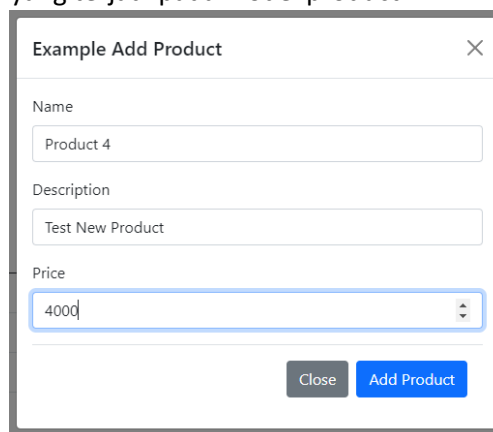
b. Membuat method store

Method *store()* pada controller *ProductController* digunakan untuk menyimpan data produk baru (Dalam studi kasus ini menambahkan product baru kedalam array). Method ini dijalankan ketika user mengirimkan form untuk membuat produk baru melalui HTTP POST request.



```
public function store(Request $request)
{
    $data = $request->all();
    $products = Product::addProduct($data);
    //return redirect()->route('products.index');
    return view('product', compact('products'));
}
```

Pada kasus nyata, method store biasanya akan me-return redirect ke halaman index (yang diberi comment), tetapi karena model kali ini menggunakan data dummy yang notabenenya disimpan dalam array, maka return adalah view dari product agar dapat terlihat perubahan yang terjadi pada model product.



The screenshot shows a web form titled 'Example Add Product'. The form has three input fields: 'Name' with the value 'Product 4', 'Description' with the value 'Test New Product', and 'Price' with the value '4000'. At the bottom of the form, there are two buttons: 'Close' and 'Add Product'.

Tampilan dari tambah product yang ada pada view product menggunakan modal.

Example table products

Click to add data

#	Name	Description	Price
1	Product 1	Description 1	1000
2	Product 2	Description 2	2000
3	Product 3	Description 3	3000
4	Product 4	Test New Product	4000

Tampilan setelah produk berhasil ditambahkan pada data dummy yang disimpan pada array.

Berikut adalah codingan lengkap dari ProductController:

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 use App\Models\Product;
5 class ProductController extends Controller
6 {
7     //Author: Akmal Muhamed Firdaus - 1301204188
8     public function index()
9     {
10         $products = Product::getAllProducts();
11         //dd($products);
12         return view('product', compact('products'));
13     }
14     public function store(Request $request)
15     {
16         $data = $request->all();
17         $products = Product::addProduct($data);
18         //return redirect()->route('products.index');
19         return view('product', compact('products'));
20     }
21     /**
22      * Show the form for creating a new resource.
23      *
24      * @return \Illuminate\Http\Response
25      */
26     public function create()
27     {
28         //
29     }
30     /**
31      * Display the specified resource.
32      *
33      * @param int $id
34      * @return \Illuminate\Http\Response
35      */
36     public function show($id)
37     {
38         //
39     }
40     /**
41      * Show the form for editing the specified resource.
42      *
43      * @param int $id
44      * @return \Illuminate\Http\Response
45      */
46     public function edit($id)
47     {
48         //
49     }
50     /**
51      * Update the specified resource in storage.
52      *
53      * @param \Illuminate\Http\Request $request
54      * @param int $id
55      * @return \Illuminate\Http\Response
56      */
57     public function update(Request $request, $id)
58     {
59         //
60     }
61     /**
62      * Remove the specified resource from storage.
63      *
64      * @param int $id
65      * @return \Illuminate\Http\Response
66      */
67     public function destroy($id)
68     {
69         //
70     }
71 }
72
```

8. Routing

Routing pada Laravel adalah mekanisme untuk menentukan bagaimana aplikasi web menangani permintaan HTTP dari client. Dalam Laravel, routing digunakan untuk menentukan alamat URL (URI) apa yang akan diarahkan ke method tertentu dalam controller.

Routing dapat digunakan untuk menangani berbagai macam metode HTTP, seperti GET, POST, PUT, PATCH, dan DELETE. Setiap metode HTTP dapat ditangani oleh sebuah route yang berbeda.

Dalam Laravel, routing dapat didefinisikan dalam file `routes/web.php` atau `routes/api.php`. Untuk setiap route, kita bisa menentukan URI, controller method yang akan menangani route tersebut, dan middleware yang akan dijalankan sebelum atau setelah route dijalankan.

Laravel juga menyediakan fitur-fitur yang memudahkan pengelolaan routing, seperti route parameters, route groups, dan named routes. Route parameters memungkinkan kita untuk menentukan parameter pada URI, seperti id atau nama, yang kemudian dapat digunakan dalam method controller. Route groups memungkinkan kita untuk mengelompokkan beberapa route yang memiliki middleware atau prefix yang sama. Named routes memungkinkan kita untuk memberikan nama pada sebuah route yang kemudian dapat digunakan dalam view atau controller.

Pada routing cukup menambahkan `Route::resource("url", namaController::class)`, maka secara otomatis routing pada laravel dapat mengenali GET, POST, PUT, PATCH, dan DELETE pada routes tersebut.

```
//Author: Akmal Muhamad Firdaus - 1301204188
Route::get('/', function () {
    return view('welcome');
});

Route::resource('products', ProductController::class);
```

Contoh lain jika ingin menggunakan method HTTP tertentu adalah pada url, yaitu kita dapat menambahkan `Route::method("url", aksi)`. Method dapat diganti sesuai dengan method apa yang ingin digunakan, lalu aksi dapat berupa mengembalikan view langsung atau memanggil salah satu function pada controller.

Berikut adalah Actions Handled By Resource ProductController

Verb/Method	URI	Action	Route Name
GET	/products	index	products.index
GET	/ products/create	create	products.create
POST	/products	store	products.store
GET	/products/{id}	show	products.show
GET	/products/{id}/edit	edit	products.edit
PUT/PATCH	/products/{id}	update	products.update
DELETE	/products/{id}	destroy	products.destory

Full Source Code : <https://github.com/codernewbie04/CI13H4-ABP-Praktikum>