Modul 9 : *Framework* Laravel (2)

9.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum adalah **170 menit,** dengan rincian sebagai berikut.

a. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
b. 60 menit untuk penyampaian materi
c. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
d. 50 menit **pengayaan**

9.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memanfaatkan Blade Templating untuk membuat halaman web.
2. Memanfaatkan model (Eloquent) untuk mengakses basis data.
3. Memanfaatkan validation untuk melakukan validasi terhadap masukan pengguna.
4. Memanfaatkan flash data untuk menampilkan pesan status.
5. Memanfaatkan authentication pada Laravel dengan *scaffolding*.

9.3 Dasar Teori

a) Blade *Templating*

Blade merupakan *template engine* yang digunakan oleh Laravel dan terinspirasi dari Razor *engine* pada .NET. Dibandingkan menggunakan PHP murni, pemanfaatan Blade sebagai template engine dapat memberikan beberapa keuntungan, yaitu:

1. *Concise syntax*
2. Mudah dipelajari
3. Intuitif
4. Mudah dikembangkan/ditingkatkan (*extensible*)
5. *Powerful*

b) Model (Eloquent)

Pada MVC, model berperan dalam menangani *business logic*. Salah satu *business logic* yang banyak dijumpai pada suatu aplikasi adalah berkaitan dengan akses dan manipulasi data. Untuk mengakses basis data, Laravel dilengkapi dengan Eloquent. Eloquent merupakan *object-relational mappers* (ORM) untuk mengatasi impedence mismatch antara objek aplikasi dengan tabel basis data sehingga dapat mempermudah dalam mengakses data.

Untuk membuat model, kita dapat menggunakan Artisan. Contohnya:

```
1   php artisan make:model Mahasiswa --migration
```

Dapat dilihat pada bagian perintah di atas, kita menambah opsi --migration. Hal ini menginstruksikan Laravel untuk membuat migrasi untuk tabel pada basis data.

c) *Validation*

Validation dapat mempermudah kita dalam memastikan validitas data yang dimasukkan oleh pengguna ke dalam sistem. Hal ini dapat mencegah masuknya data-data yang tidak valid atau bahkan data yang dapat membahayakan sistem.

d) *Flash Data*

Flash data dapat digunakan untuk menyimpan data dalam waktu singkat, yaitu hanya 1 kali digunakan saja. Ketika flash data telah diakses, maka dia akan hilang pada pengaksesan berikutnya. *Flash data* termasuk dalam kategori *session data*. Flash data akan tersedia untuk request berikutnya (misal dari halaman A melakukan *redirect* menuju ke halaman B, jika halaman A menghasilkan *flash data*, maka data akan tersedia untuk halaman B pada *request* tersebut saja) setelah itu akan dihapus.

e) **Authentication**

Laravel telah dilengkapi fitur untuk menghasilkan authentication secara lengkap dengan metode *scaffolding*. Fitur authentication yang dihasilkan meliputi antara lain:

1. Registrasi
2. Login
3. Logout
4. Lupa password
5. Email konfirmasi

9.4 Prosedur Praktikum

1. Struktur Direktori View

a. Buatlah sebuah proyek baru Laravel dengan perintah sbb:

```
laravel new praktikumLaravel2
```

b. Pada folder `resources/views`, buatlah direktori baru dengan nama praktikum2. Kemudian pada direktori tersebut buatlah sebuah file Blade dengan nama `index.blade.php` dan isi dengan kode sbb:

```
<!doctype html>
<html>
    <head>
        <title>Praktikum Laravel Ke-2</title>
    </head>
    <body>
        <h1>Praktikum Laravel Ke-2</h1>
    </body>
</html>
```

c. Pada file `web.php` dalam folder `routes`, tambahkan rute sbb:

```
Route::get('/praktikum-laravel', function () {
    return view('praktikum2.index');
});
```

d. Apakah URL untuk mengakses halaman tersebut?

e. Mengapa pada *method* view menggunakan parameter 'praktikum2.index'?

f. Jalankan halaman tersebut, tempelkan *screenshot*-nya pada tempat yang telah disediakan di bawah ini.

2. Perulangan dengan Sintaks Blade

a. Ubahlah isi *file* `index.blade.php` menjadi sbb:

```
<!doctype html>
<html>
    <head>
        <title>Praktikum Laravel Ke-2</title>
    </head>
    <body>
        <ul>
            <li>Ini adalah urutan ke-1</li>
            <li>Ini adalah urutan ke-2</li>
            <li>Ini adalah urutan ke-3</li>
            <li>Ini adalah urutan ke-4</li>
            <li>Ini adalah urutan ke-5</li>
        </ul>
    </body>
</html>
```

b. Kemudian ubahlah lagi menggunakan *loop* dengan sintaks Blade. Tempelkan kode yang saudara buat pada tempat yang telah disediakan di bawah ini:

3. Menampilkan Data Dari *Route Parameter*

a. Buatlah sebuah *controller* baru dengan perintah sbb:

```
1  php artisan make:controller EventsController
```

b. Buka *file controller* yang baru saja di-*generate* pada folder `app\Http\Controllers` dan tambahkan sebuah *method* pada *class controller* tersebut seperti kode berikut:

```
1  public function show($location, $name)
2  {
3      return view('events.show')
4          ->with('location', $location)
5          ->with('name', $name);
6  }
```

c. Tambahkan rute berikut:

```
1  Route::get('events/{location}/{name}', [App\Http\Controllers\EventsController::class,
2  'show']);
```

d. Apakah nama file *view* (Blade) yang harus dibuat?

e. Pada direktori apakah *file view* tersebut harus diletakkan?

f. Tuliskan sebuah contoh URL untuk mengakses halaman tersebut.

g. Buatlah file *view* (Blade) tersebut sehingga dapat menampilkan nilai variabel `location` dan `name`. Tempel kode yang dihasilkan pada tempat yang telah disediakan di bawah ini:

4. Model Untuk Menampilkan Data Dari Database

a. Jalankan MySQL dan buatlah database baru dengan nama "`praktikumlaravel2`". (Bila saudara menggunakan XAMPP, jalankan Apache dan MySQL kemudian klik tombol Admin pada baris MySQL untuk membuka aplikasi phpMyAdmin. Melalui phpMyAdmin, saudara bisa membuat database baru tersebut).

b. Pada file `.env` yang berada pada direktori *root* proyek Laravel, ubahlah nama database MySQL sesuai nama yang tadi telah dibuat (jika namanya berbeda). Bila diperlukan, atur pula username, password atau konfigurasi lainnya untuk mengakses MySQL.

c. Buatlah model dengan menjalankan perintah sbb:

```
1  php artisan make:model Event --migration
```

d. Buka *file migration* yang di-*generate* oleh Artisan pada direktori "`/database/migration`" pilih file yang sesuai dengan model yang baru saja dibuat (nama *file* berakhiran `...._create_events_table`). Kemudian ubah method `up()` menjadi sbb:

```php
public function up()
{
    Schema::create('events', function (Blueprint $table) {
        $table->increments('id');
        $table->string('location');
        $table->string('name');
    });
}
```

e. Lakukan migrasi dengan menjalankan perintah sbb:

```
php artisan migrate
```

f. Tempel *screenshot* CMD atau Terminal setelah perintah di atas dijalankan pada tempat yang disediakan di bawah ini.

g. Isi tabel `events` dengan data sebagai berikut (Saudara bisa menggunakan aplikasi phpMyAdmin atau sejenisnya):

id	location	name
1	Malang	Kerja Bakti
2	Banjarmasin	Lomba Melukis
3	Medan	Rapat Tahunan

h. Pada *class* `EventsController`, tambahkan method `index()` sbb:

```php
public function index()
{
    $events = Event::all();
    return view('events.index')->with('events', $events);
}
```

i. Pada kode `EventsController.php`, tambahkan pula pernyataan untuk *import* sbb:

```php
use App\Models\Event;
```

j. Kemudian buatlah view dengan lokasi dan nama sesuai pada method `index()` di atas, isi dengan kode sbb:

```html
<!doctype html>
<html>
    <head>
        <title>Events - Praktikum Laravel Ke-2</title>
    </head>
    <body>
        <h1>Events</h1>
```

```
        <ul>
            @forelse($events as $event)
                <li>{{ $event->location }} - {{ $event->name }}</li>
            @empty
                <li>Belum ada event.</li>
            @endforelse
        </ul>
    </body>
</html>
```

k. Tambahkan rute berikut:

```
Route::get('events', [App\Http\Controllers\EventsController::class, 'index']);
```

l. Buka halaman tersebut pada *web browser* dan tempelkan *screenshot*-nya pada tempat yang disediakan di bawah ini:

5. Flash Message Untuk Menampilkan Pesan

a. Tambahkan rute berikut:

```
Route::get('events/validate',[App\Http\Controllers\EventsController::class,
'showValidateForm'])->name('validateform.event');

Route::post('events/validate',[App\Http\Controllers\EventsController::class,
'validateForm'])->name('validate.event');
```

b. Pada folder `view`, buatlah *file* baru bernama "`flash-message.blade.php`", kemudian isi dengan kode sbb:

```
@if ($message = Session::get('success'))
    <strong>{{ $message }}</strong>
@endif

@if ($errors->any())
    <strong>Data tidak valid</strong>
@endif
```

c. Pada *class* `EventsController`, tambahkan dua *method* sebagai berikut:

```
public function showValidateForm()
{
    return view('events.validate');
}

public function validateForm(Request $request)
{
    $request->validate([
       'location' => 'required|alpha|min:3|max:50',
       'name' =>  'required|string|min:3|max:50'
    ]);

    return back()->with('success','Data valid');
}
```

d. Jelaskan validasi apa saja yang dilakukan pada data `location` dan `name`.

e. Sesuai isi *method* `showValidateForm` yang baru saja ditambahkan, buatlah *file* Blade dengan isi sbb:

```
<!doctype html>
<html>
    <head>
        <title>Event Validation</title>
    </head>
    <body>
        @include('flash-message')
        <form action="{{ route('validate.event') }}" method="post">
            @csrf
            Location: <input type="text" name="location"><br>
            Name: <input type="text" name="name"><br>
            <input type="submit" value="Validate">
        </form>
    </body>
</html>
```

f. Pada formulir di atas, apakah fungsi `@csrf`? Apa yang terjadi jika baris tersebut (`@csrf`) dihapus?

g. Jalankan aplikasi, tempelkan tiga buah *screenshot*, yaitu: *screenshot* formulir validasi, *screenshot* ketika data tidak valid, dan *screenshot* ketika data valid.

6. Menambahkan Authentication dengan Breeze

a. Pastikan MySQL Server dalam keadaan berjalan.

b. Buatlah salinan *file* rute (`web.php`) untuk mengantisipasi hilangnya rute-rute yang telah didefinisikan sebelumnya.

c. Buka CMD/Terminal dan jalankan perintah-perintah berikut secara berurutan pada *project root directory*:

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install
npm run dev
php artisan migrate
```

d. Buka file `web.php` pada folder `routes`, periksa apakah rute-rute yang sebelumnya dibuat telah hilang atau masih ada. Bila hilang, maka tambahkan kembali rute-rute yang hilang tersebut.

e. Jalankan aplikasi, lakukan registrasi pengguna baru melalui path `/register`.

f. Tempelkan *screenshot* halaman Dashboard yang tampil setelah proses registrasi berhasil.

g. Lakukan proses *logout* melalui menu yang ada pada halaman Dashboard tersebut.

h. Selanjutnya kita akan melindungi halaman validasi yang sebelumnya telah kita buat (`validateform.event` dan `validate.event`) sehingga hanya bisa diakses oleh pengguna yang sedang *logged in*. Sisipkan kode yang bertanda khusus berikut ini pada rute:

```
Route::get('events/validate',[App\Http\Controllers\EventsController::class,
'showValidateForm'])->middleware(['auth'])->name('validateform.event');
Route::post('events/validate',[App\Http\Controllers\EventsController::class,
'validateForm'])->middleware(['auth'])->name('validate.event');
```

i. Dalam posisi tidak *logged in*, akses halaman formulir validasi. Apa yang terjadi?

j. Dalam posisi *logged in*, akses halaman formulir validasi. Apa yang terjadi?