id-laravel.com

Membuat Form Registrasi Dengan Laravel

Salah satu prinsip ketika membuat aplikasi adalah: **mulailah dengan tampilan**. Dengan tampilan, klien Anda akan lebih memahami prototype aplikasi yang Anda usulkan. Dari tampilan, kita bisa lebih mudah mendesain skema basis data. Oleh karena itu, mari kita mulai membuat form sederhana dengan laravel.

Agar lebih menarik, pura-puranya kita akan membuat sebuah aplikasi social network khusus untuk pengangguran. Dengan jumlah pengangguran yang mencapai jutaan, tentunya aplikasi ini memiliki prospek yang sangat cerah di Indonesia. Untuk tahap awal, kita akan membuat sebuah form registrasi dengan field-field beserta validasinya sebagai berikut:

1. **Email** (sekaligus sebagai username, harus alamat email yang valid, dan harus unik)
2. **Password** (minimal 6 huruf)
3. **Konfirmasi password** (harus sama dengan password)

Cukup sederhana kan formnya? Selanjutnya kita bisa mulai koding. Jika ingin ke kamar kecil atau ingin membuat kopi terlebih dahulu, lakukan sekarang karena setelah ini kita akan masuk ke mode

serius. Yup, pernyataan saya bahwa membuat social network untuk pengangguran memiliki prospek cerah hanya bercanda belaka dan tidak ada dasar hukumnya :D

Langkah 1: Routing

Seperti biasa, mau bikin apapun selalu dimulai dari `routing`. Buka file `app/routes.php` dan tambahkan kode berikut:

```
Route::get('/register', function()
{
    return View::make('register');
}); Jadi nanti kita bisa mengakses form registrasi melalui `http://localhost/laravel/public/register` (atau sesuai dengan penamaan folder Anda sendiri).
```

Langkah 2: View

Karena framework laravel dibuat dengan keindahan, maka form yang akan kita bikin juga harus terlihat indah. Untuk itu diperlukan usaha untuk styling menggunakan CSS. Tapi karena saya terlalu malas untuk membuat CSS sendiri, maka kita gunakan yang sudah ada saja, bukan bootstrap, melainkan kube css. Kenapa kube? Karena lebih simpel, sederhana, dan tidak begitu terkenal.

Silakan kunjungi situs imperavi.com/kube/ dan download filenya. Ekstrak file hasil download, lalu cari file `kube.min.css` dan copy ke folder `public/css`. Jika belum ada folder `css`, silakan dibuat

sendiri.

Selanjutnya adalah membuat view, tambahkan file baru `app/views/register.php` dan tuliskan kode berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>Title</title>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css"
href="<?php echo asset('css/kube.min.css') ?>" />

</head>
<body style="width:80%;margin:40px auto">
    <div class="units-container">

        <form method="post" action=""
class="forms">
            <h3>Form Registrasi</h3>
            <label>
                Email <span class="error"><?php
echo $errors->first('email') ?></span>
                <input type="text" name="email"
value="<?php echo Form::old('email') ?>"
```

```
class="width-50" />
            </label>
            <label>
                Password <span
class="error"><?php echo
$errors->first('password') ?></span>
                <input type="password"
name="password" value="<?php echo
Form::old('password') ?>" class="width-50" />
            </label>
            <label>
                Password Confirmation <span
class="error"><?php echo
$errors->first('password_confirmation') ?></span>
                <input type="password"
name="password_confirmation" value="<?php echo
Form::old('password_confirmation') ?>"
class="width-50" />
            </label>
            <input type="submit" class="btn"
value="Submit">

        </form>

    </div>
</body>
</html>
```

Tidak ada yang aneh kan, hanya halaman html biasa berisi form registrasi dengan sedikit styling dengan meng-include satu file css. Jika diperhatikan di view tersebut Anda bisa melihat ada beberapa fungsi php untuk meng-generate elemen form, diantaranya `Form::text()` dan `Form::select()`.

Meskipun baru pertama kali melihat fungsi-fungsi tersebut, saya yakin Anda sudah bisa menebak apa fungsinya. Untuk `Form::text()` parameter pertama adalah nama elemen form, parameter kedua untuk mengeset value, dan parameter ketiga adalah atribut tambahan untuk elemen form tersebut. Oia, `Input::old()` digunakan untuk mendapatkan isian form yang pernah disubmit sebelumnya. Untuk lebih lengkapnya silakan baca-baca dokumentasi resmi laravel untuk Form.

Oia, jika Anda berhasil seharusnya tampilan formnya seperti ini:
Tampilan Form Registrasi

Langkah 3: Persiapkan database

Dengan tampilan seperti di atas, maka kita perlu mempersiapkan tabel untuk menyimpan data user yang melakukan registrasi. Skema yang digunakan adalah sebagai berikut:

```
CREATE TABLE `users` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `created_at` datetime DEFAULT NULL,
```

```
  `updated_at` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT
CHARSET=latin1;
```

> By default, kolom **created_at** dan **updated_at** dibutuhkan laravel untuk menyimpan informasi waktu kapan data disimpan dan dimodifikasi di database.

Langkah 4: Submit form

Jika Anda men-submit form tersebut akan ditampilkan pesan error. Hal ini dikarenakan kita belum menambahkan `routing`. Ingat aturannya, mau menambahkan halaman apapun selalu dimulai dengan `routing`.

Ok, sampai sini Anda mungkin bertanya-tanya, bukankah jika ada form dengan atribut `action=""` maka berarti form tersebut akan disubmit ke url yang sama. Lalu bukanlah sebelumnya kita sudah membuat `routing` untuk halaman register? Lalu dimana letak errornya?

Letak errornya adalah pada langkah 1 di atas, dimana kita mendaftarkan url `/register` menggunakan fungsi `Route::get()`. Kata kuncinya ada di `get()`, artinya routing tersebut hanya berlaku jika halaman register di-request menggunakan method GET (misalnya dibuka melalui browser). Sedangkan ketika kita submit form, method yang digunakan adalah POST.

Oleh karena itu kita perlu menambahkan satu `routing` lagi untuk menangani POST seperti berikut ini:

```
Route::post('/register', function()
{
    var_dump($_POST);
});
```

Isi kembali form registrasi Anda lalu submit. Anda bisa lihat kita sudah berhasil mendapatkan data yang diisikan di form.

Langkah 5: Simpan ke database menggunakan Model

Pada tulisan sebelumnya tentang interaksi dengan database, sudah dijelaskan bahwa salah satu cara untuk menyimpan data ke database adalah dengan menggunakan `Eloquent` ORM. Untuk menggunakan `Eloquent` ORM terlebih dahulu kita harus membuat `Model`. Singkatnya, karena form yang kita gunakan adalah untuk menyimpan data user, maka kita perlu membuat model `User`.

Buka folder `app/models` dan simsalabim... Anda akan menemukan file `User.php` yang isinya kelas untuk `Model User`. File tersebut adalah file bawaan dari laravel yang ditujukan untuk keperluan autentikasi user.

Karena saat ini kita belum membutuhkan autentikasi user, jadi abaikan saja isi file tersebut dan timpa dengan kode yang lebih sederhana seperti ini:

```
<?php

class User extends Eloquent{



}
```

Kode di atas berarti kita sudah mendefinisikan sebuah model (ditandai dengan `extends Eloquent`) dengan nama `User`. Selanjutnya kita ubah kode di `routes.php` untuk menangani submission form seperti ini:

```
Route::post('/register', function()
{
    $user = new User;
    $user->email    = Input::get('email');
    $user->password =
Hash::make(Input::get('password'));
    $user->save();

    return Redirect::to("register");
});
```

Isi kembali form registrasi kemudian submit. Setelah itu perhatikan tabel users di database Anda dan lihat apakah data sudah tersimpan.

> Fungsi `Hash::make()` berguna untuk melakukan hashing password. Mirip seperti `md5()` yang biasa kita pake di PHP, tetapi

> jauh lebih aman.

Langkah 6: Validasi form

Langkah selanjutnya adalah melakukan validasi sesuai aturan yang sudah kita sepakati di bagian awal tulisan ini. Untuk keperluan tersebut, maka kode di atas perlu kita modifikasi lagi:

```
Route::post('/register', function()
{
    // 1. setting validasi
    $validator = Validator::make(
        Input::all(),
        array(
            "email"                 =>
"required|email|unique:users,email",
            "password"              =>
"required|min:6",
            "password_confirmation" =>
"same:password",
        )
    );

    // 2a. jika semua validasi terpenuhi simpan
ke database
    if($validator->passes())
    {
        $user = new User;
        $user->email    = Input::get('email');
```

```
            $user->password =
Hash::make(Input::get('password'));
            $user->save();

            return Redirect::to("register");
        }
        // 2b. jika tidak, kembali ke halaman form
registrasi
        else
        {
            return Redirect::to('register')
                ->withErrors($validator)
                ->withInput();
        }
});
```

`Validator::make()` adalah fungsi untuk membuat aturan validasi, parameter pertama adalah data yang akan divalidasi (biasanya adalah $_POST atau di laravel biasa menggunakan Input::all()), parameter kedua adalah aturan validasi yang ingin diterapkan terhadap data tersebut. Daftar lengkap aturan validasi yang sudah disediakan laravel bisa dilihat disini.

Selanjutnya kita memanggil fungsi $validator->passes() untuk mengecek apakah semua aturan validasi sudah terpenuhi. Jia iya maka data tersebut kita simpan ke database. Jika tidak, kita redirect kembali ke halaman form registrasi, dengan tetap menyimpan informasi field mana yang validasinya gagal

(menggunakan fungsi `->withErrors($validator)`) serta informasi tentang value form yang sudah diisikan sebelumnya (menggunakan fungsi `->withInput()`).

Coba buka kembali form registrasi Anda, kosongkan beberapa field atau isi dengan data yang salah, maka akan ditampilkan pesan error seperti di bawah ini:

Pesan error pada form

Terakhir, coba isi form tersebut dengan data yang benar, maka pesan error tidak akan ditampilkan dan data akan disimpan ke database.

Bonus: Menampilkan pesan sukses

Salah satu ciri aplikasi yang baik adalah selalu memberikan feedback kepada user terhadap semua aksi yang dilakukan. Kita sudah menampilkan pesan error ketika user mengisi form dengan tidak benar, maka seharusnya kita juga menampilkan pesan sukses ketika data yang diisi sudah benar dan berhasil disimpan ke database.

Mari kita modifikasi sedikit kode untuk tampilan form registrasi menjadi seperti berikut:

```
<?php if(Session::has('register_success')): ?>
    <div class="message message-success">
        <span class="close"></span>
        <?php echo
```

```
Session::get('register_success') ?>
    </div>
<?php endif; ?>
```

Yup, sesuai namanya, kita melakukan pengecekan terhadap `Session` untuk menentukan apakah pesan sukses ditampilkan atau tidak. `Session` tersebut perlu diset dari bagian yang menangani submission form seperti di bawah ini:

```
...
if($validator->passes())
{
    $user = new User;
    $user->email    = Input::get('email');
    $user->password =
Hash::make(Input::get('password'));
    $user->save();

    return
Redirect::to("register")->with('register_success',
'Selamat, Anda telah resmi menjadi pengangguran,
silakan cek email untuk aktivasi :P');
}
...
```

Kita memanggil fungsi `->with('key', 'value')` untuk mengeset `session` yang nantinya akan digunakan di bagian `view`.

Isi kembali form registrasi dengan benar, lalu submit. Jika Anda

sehati dan sekodingan dengan saya, maka seharusnya akan ditampilkan pesan sukses seperti di bawah ini:

Pesan sukses pada form

Selamat, Anda sudah berhasil membuat halaman registrasi menggunakan laravel :)