

Model

Pendahuluan

Pada praktikum kali ini anda akan mempelajari tentang konsep model pada framework Laravel termasuk koneksi ke database, schema migrations, seeder, dan caching.

Tujuan Pembelajaran

1. Mahasiswa mampu memahami konsep model pada Web Framework
2. Mahasiswa mampu membuat koneksi ke database
3. Mahasiswa mampu membuat schema migrations
4. Mahasiswa mampu membuat seeder

Alat dan Bahan

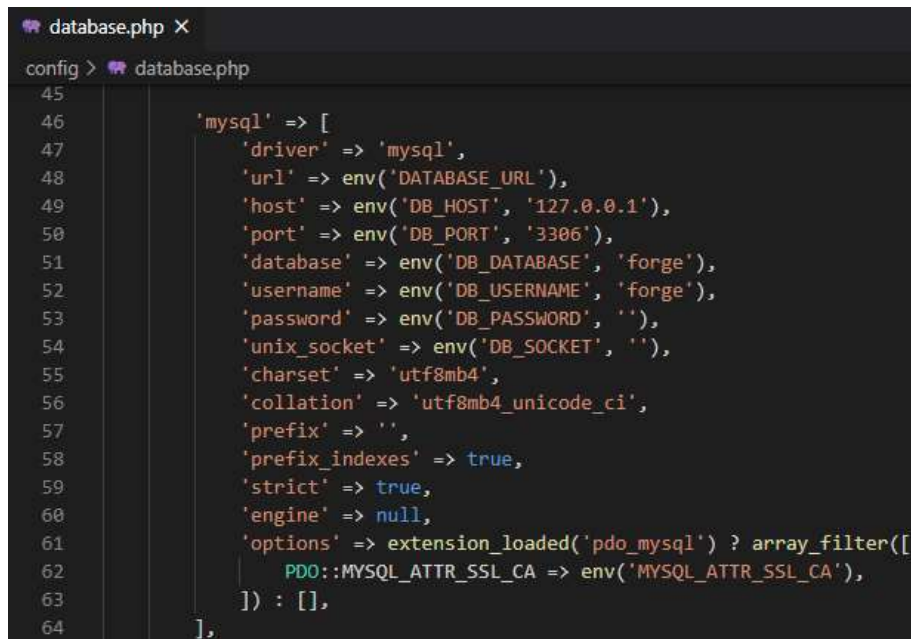
1. PC atau Laptop
2. Text Editor/IDE (rekomendasi PHPStorm atau VSCode)
3. Web Browser
4. PHP

Model

Model adalah bagian dari MVC yang bertanggung jawab untuk menangani representasi basis data (database). Model bertugas untuk melakukan query ke database, insert data baru, update, atau hapus record di database. Laravel menggunakan Eloquent, sebuah ORM yang memudahkan interaksi tabel database dengan Model yang berhubungan.

Konfigurasi Koneksi Database

Untuk dapat bekerja dengan basis data, sebelumnya dibutuhkan proses konfigurasi koneksi ke basis data. Nilai konfigurasi yang dibutuhkan didefinisikan pada file `config/database.php`, nilai yang dikonfigurasi bergantung pada jenis engine basis data yang digunakan.

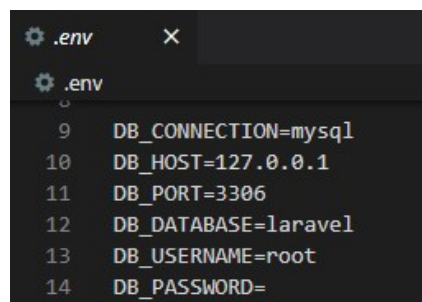


```

45
46     'mysql' => [
47         'driver' => 'mysql',
48         'url' => env('DATABASE_URL'),
49         'host' => env('DB_HOST', '127.0.0.1'),
50         'port' => env('DB_PORT', '3306'),
51         'database' => env('DB_DATABASE', 'forge'),
52         'username' => env('DB_USERNAME', 'forge'),
53         'password' => env('DB_PASSWORD', ''),
54         'unix_socket' => env('DB_SOCKET', ''),
55         'charset' => 'utf8mb4',
56         'collation' => 'utf8mb4_unicode_ci',
57         'prefix' => '',
58         'prefix_indexes' => true,
59         'strict' => true,
60         'engine' => null,
61         'options' => extension_loaded('pdo_mysql') ? array_filter([
62             PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
63         ]) : [],
64     ],

```

Contoh: Ketika Laravel akan dihubungkan dengan MySQL, maka konfigurasi yang dibutuhkan dapat dilihat pada data array 'mysql' seperti pada gambar di atas. Perhatikan pada baris yang memanggil fungsi helper `env()`. Helper ini membutuhkan setidaknya dua parameter yaitu berupa key dan nilai default. Nilai-nilai key didefinisikan pada file `.env` yang terletak di root direktori project. Contoh: pada `env('DB_HOST', '127.0.0.1')` maksudnya adalah nilai `DB_HOST` didefinisikan sesuai dengan nilai pada `.env`. Jika tidak ada definisi maka nilainya adalah `127.0.0.1`.



```

9  DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=laravel
13 DB_USERNAME=root
14 DB_PASSWORD=

```

Sehingga untuk mengkonfigurasi basis data MySQL dibutuhkan pengaturan nama host, port, nama database, username, dan password dari database seperti tertera pada gambar di atas. Konfigurasi ini dilakukan pada file `.env` yang kemudian nilainya dibaca dalam `config/database.php`.

Eloquent

Model umumnya berada pada direktori app\Models (mulai Laravel 8, versi sebelumnya Model berada pada direktori app). Untuk membuat model digunakan perintah make:model pada perintah Artisan. Berikut contoh pembuatan model bernama Post.

```
php artisan make:model Post
```

Jika dibutuhkan pembuatan migration yang dikaitkan dengan model, proses pembuatan model dapat menambahkan opsi --migration atau -m.

```
php artisan make:model Post --migration
```

Selain migration, proses pembuatan model terkadang melibatkan factory, seeder dan juga controller. Beberapa variasi perintah yang memungkinkan dapat dilihat pada kode berikut.

```
php artisan make:model Post --factory
php artisan make:model Post -f

php artisan make:model Post --seed
php artisan make:model Post -s

php artisan make:model Post --controller
php artisan make:model Post -c

php artisan make:model Post -mfsc
```

Model yang dibuat merupakan turunan dari Eloquent Model. Dalam Eloquent terdapat beberapa ketentuan yang harus dipahami terkait penamaan tabel. Sebagai contoh pada tabel berikut terdapat model Post. Eloquent mengasumsikan model tersebut berasosiasi dengan tabel posts. Secara umum penamaan model menggunakan kata tunggal, dan tabel yang berasosiasi menggunakan kata jamak.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
}
```

Jika dibutuhkan adanya penamaan di luar aturan tersebut, dapat didefinisikan secara manual dengan mendefinisikan property `$table` dengan akses `protected` pada model. Selain nama tabel, Eloquent juga mengasumsikan kolom primary key dari suatu model dinamakan `id`. Jika dibutuhkan, definisi primary key dapat diubah dengan mendefinisikan property `$primaryKey`. Sebagai tambahan, Eloquent juga mengasumsikan bahwa primary key dari model bersifat auto increment. Jika dibutuhkan primary key berupa non-increment, maka dapat didefinisikan property `$incrementing` dengan nilai `false`. Dan juga untuk definisi tipe non-numeric dapat didefinisikan dengan property `$keyType`. Definisi lebih lanjut dapat dibaca pada dokumentasi Laravel <https://laravel.com/docs/8.x/eloquent#eloquent-model-conventions>

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    protected $table = 'my_posts';
    protected $primaryKey = 'my_id';
    protected $keyType = 'string';
    public $incrementing = false;
}
```

Dengan penggunaan Eloquent, dapat dilakukan proses pengambilan data dari database dengan menggunakan Model yang berasosiasi. Method `all` akan melakukan pengambilan semua data yang ada dalam tabel database. Berikut beberapa contoh kode untuk pengambilan data:

<code>Post::all()</code>	Mengambil keseluruhan data pada tabel
<code>Post::where('slug', 'about-us')->first()</code>	Mengambil data pertama yang ditemukan dengan slug bernilai about-us
<code>Post::paginate(5)</code>	Tampilkan data sebanyak 5 untuk setiap halaman (pagination)

Migration

Migration ibaratnya seperti version control untuk database. Hal ini memberikan kemudahan suatu skema database didefinisikan dan dibagikan dalam suatu tim. Sehingga versi skema database untuk masing-masing orang akan selalu sama.

Pembuatan migration menggunakan perintah `Artisan make:migration`. File migration akan diletakkan dalam direktori `database/migrations`. Setiap nama file migration akan ditambahkan informasi timestamp. Informasi tersebut digunakan Laravel untuk menentukan urutan proses migrasi skema.

```
php artisan make:migration create_posts_table
```

Struktur suatu migration memiliki dua fungsi up dan down. Fungsi up digunakan untuk membuat tabel baru, menambah kolom atau indeks ke dalam database. Sedangkan fungsi down digunakan untuk mengembalikan operasi yang dilakukan oleh fungsi up. Sebagai gambaran pada kode berikut terdapat tabel posts yang memiliki atribut id, title, slug, content, dan draft. Terdapat atribut bawaan dari laravel yaitu created_at dan updated_at yang diwakili dengan fungsi timestamps() pada migrasi.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title', 100)->index();
            $table->string('slug', 100)->index();
            $table->text('content');
            $table->text('image');
            $table->boolean('draft')->default(false);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('posts');
    }
}
```

Migrasi dapat dijalankan dengan menggunakan perintah Artisan migrate. Perintah Artisan migrate memiliki beberapa jenis, antara lain: status, rollback, refresh dan lain-lain. Silahkan baca dokumentasi Laravel pada tautan <https://laravel.com/docs/8.x/migrations#running-migrations>

```
php artisan migrate
```

Seeding

Dalam konteks pengembangan aplikasi, **seeding** adalah memberikan data awal ke database. Hal ini biasanya dilakukan saat pengembangan terutama jika dibutuhkan apakah fitur tertentu telah berjalan sesuai ekspektasi. Sebenarnya dapat dilakukan pemberian data awal secara manual dengan melakukan insert melalui DBMS, akan tetapi kurang praktis jika butuh menguji banyak data. Oleh karena itu dibutuhkan pemanfaatan fitur Seeding di Laravel. Untuk membuat seeder, digunakan perintah Artisan make:seeder. File seeder akan diletakkan pada direktori database/seeders.

```
php artisan make:seeder PostSeeder
```

Sebuah seeder berisi sebuah fungsi bernama run. Fungsi tersebut akan dipanggil ketika perintah Artisan db:seed dieksekusi. Dalam blok fungsi tersebut, berisi logika pengisian data ke dalam database sesuai yang dibutuhkan. Perhatikan ilustrasi kode berikut untuk memahami sebuah class Seeder.

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PostSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('posts')->insert([
            'title' => 'Belajar Model Dengan Laravel',
            'slug' => ' belajar-model-dengan-laravel',
            'image' =>
'https://via.placeholder.com/640x480.png/00ff77?text=Belajar Model',
            'content' => 'Belajar Laravel itu menyenangkan',
            'draft' => 0
        ]);
    }
}
```

```
        ]);  
    }  
}
```

Untuk menjalankan sebuah seeder, dapat dilakukan dengan menggunakan perintah Artisan db:seed diikuti parameter nama class Seeder. Sebagai contoh silahkan lihat ilustrasi berikut.

```
php artisan db:seed --class=PostSeeder
```

Jika terdapat banyak seeder yang dibutuhkan, dapat didefinisikan proses pemanggilan pada class DatabaseSeeder. Sebagai ilustrasi, perhatikan contoh berikut.

```
public function run()  
{  
    $this->call(UserSeeder::class);  
    $this->call(PostSeeder::class);  
}
```

Kemudian jalankan perintah berikut untuk mengeksekusi proses seed.

```
php artisan db:seed
```

Selain pendefinisian data secara manual, dapat juga digunakan konsep factory untuk memudahkan pembuatan data uji coba. Untuk membuat factory, dapat menggunakan perintah Artisan make:factory. Factory menggunakan library Faker untuk membuat data uji coba. Pada factory didefinisikan format data yang berasosiasi dengan kolom pada tabel database. Dokumentasi terkait jenis format yang didukung dapat dilihat pada halaman GitHub Faker <https://github.com/fzaninotto/Faker>. Sebagai ilustrasi penggunaan factory dalam seeder, silahkan perhatikan kode berikut:

```
php artisan make:factory PostFactory
```

```
<?php  
  
namespace Database\Factories;  
  
use App\Models\Post;  
use Illuminate\Database\Eloquent\Factories\Factory;  
  
class PostFactory extends Factory  
{  
    /**  
     * The name of the factory's corresponding model.  
     *  
     * @var string  
     */  
    protected $model = Post::class;  
  
    /**  
     * Define the model's default state.  
     */  
}
```

```

        *
        * @return array
        */
    public function definition()
    {
        $title = $this->faker->sentence;
        $slug = str_slug($title);
        return [
            'title' => $title,
            'slug' => $slug,
            'image' => $this->faker->imageUrl(640, 480, 'animals',
true),
            'content' => $this->faker->realText(),
            'draft' => random_int(0, 1),
        ];
    }
}

```

Pemanggilan factory dapat dilakukan pada class Seeder atau DatabaseSeeder. Silahkan pelajari pada dokumentasi Laravel <https://laravel.com/docs/8.x/seeding#using-model-factories>. Selain itu, model yang berasosiasi dengan factory harus ditambahkan dengan trait HasFactory pada definisi Eloquent Model.

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;
}

```

Praktikum

Praktikum 1 – Menganalisa Studi Kasus

1. Pelajari contoh studi kasus penerapan model pada repository kode <https://github.com/dhanifudin/belajar-model>
2. Clone repository, install dan jalankan sampai project tersebut bisa dijalankan

Praktikum 2 – Menghubungkan Model

1. Lanjutkan pengerjaan praktikum pada pertemuan 3. Terapkan konsep model pada project Laravel yang telah menggunakan template Bootstrap. Data yang ditampilkan pada halaman diambil dari database.
2. Kerjakan secara berkelompok dan lakukan commit untuk setiap proses yang dibutuhkan.