TUJUAN PRAKTIKUM

1. Mahasiswa mampu memahami interaksi database dengan framework
2. Mahasiswa mampu mengetahui penggunaan eloquent dalam interaksi database

TEORI SINGKAT

Eloquent ORM (Object Relational Mapping) merupakan teknik untuk memetakan basis data relasional ke model objek. Berinteraksi dengan database seperti menampilkan, menambah, mengubah atau menghapus data menggunakan Eloquent ORM lebih disarankan walaupun kita dapat menggunakan Query Builder tanpa membuat model terlebih dahulu.

Hal yang perlu diperhatikan sebelum menggunakan Qloquent ORM untuk berintaksi dengan database adalah model, secara default eloquent akan menganggap nama class model adalah sebagai nama tabel dengan menambahkan "s" dibelakangnya. Secara default eloquent juga akan beramsumi kolom primary key dari tabel akan bernama id, serta eloquent juga kan beramsumi kita ingin mengetahui kapan suatu data ditambah dan diperbaharui maka dari itu kita perlu menyiapkan dua kolom tambahan yang bernama **created_at** dan **updated_at** yang bertipe datetime. Pada contoh ini kita akan merubah semua struktur tabel yang telah kita buat sebelumnya. Hapus semua tabel yang ada pada database **websaya,** kemudian hapus juga file ***_ubah_tabel_mahasiswa.php** dan perbaharui file migration ***_create_mahasiswa_table.php** seperti berikut:

```php
public function up()
{
    Schema::create('mahasiswa', function (Blueprint $table) {
        $table->integer('nim')->primary();
        $table->string('nama');
        $table->string('prodi');
        $table->year('angkatan');
        $table->dateTime('created_at');
        $table->dateTime('updated_at');
    });
}
```

Selanjunya perbaharui kembali ***_create_prodi_table.php** dengan perintah sebagai berikut:

```php
public function up()
{
    Schema::create('prodi', function (Blueprint $table) {
        $table->string('kd_prodi')->primary();
        $table->string('nama_prodi');
        $table->dateTime('created_at');
        $table->dateTime('updated_at');

    });
}
```

Pertama-tama periksalah model dari tabel prodi pada folder app/ jika model belum dibuat maka buatlah menggunakan perintah artisan make:model atau seperti berikut:

```
php artisan make:model prodi
```

perintah tersebut akan menghasilkan file model baru yang bernama prodi.php pada folder app/. Bukalah dan modifikasi file model **prodi.php** tersebut menjadi seperti berikut:

```php
<?php

namespace App\Models;

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

class prodi extends Model
{

    public $table = "prodi";
}
```

Pada script diatas kita mendefinisikan satu variabel bernama table dengan nilai variabel tersebut adalalah prodi, hal ini dilakukan karena tabel dengan nama prodi yang telah kita buat tidak memenuhi syarat nama tabel dengan menggunakan

ELOQUENT maka dari itu kita bisa mengakalinya dengan mendefinisikan variabel tersebut di dalam model.

DATA DENGAN ELOQUENT ORM

Untuk menampilkan data kategori produk dengan ELOQUENT pertama-tama kita harus membuat sebuah route pada file **web.php** yang ada pada folder **routes/web.php** dengan skript sebagai berikut:

```
use App\Http\Controllers\prodiController;
Route::get('/prodi', [prodiController::class,'index']);
Route::get('/prodi/create', [prodiController::class,'create']);
Route::get('/prodi/update', [prodiController::class,'update']);
Route::get('/prodi/destroy', [prodiController::class,'destroy']);
```

Pada route diatas jika kita mengakses link http://localhost:8000/prodi maka kita akan diarahkan ke fungsi **index()** yang ada pada file prodiController.php. Jadi karena controller prodiController.php belum dibuat maka buatlah dengan menggunakan perintah artisan berikut pada cmd atau comand prompt:

```
php artisan make:controller prodiController
```

selanjutnya bukalah file prodiController.php yang berada pada folder app\Http\Controller dan pada bagian atas controller tambahkan script **`use App\Models\prodi;`** menambahkan skript tersebut berarti kita telah menambakan class model prodi ke dalam class controller prodiController yang artinya untuk berinteraksi dengan tabel kateogri kita hanya perlu memanggil nama dari modelnya saja.

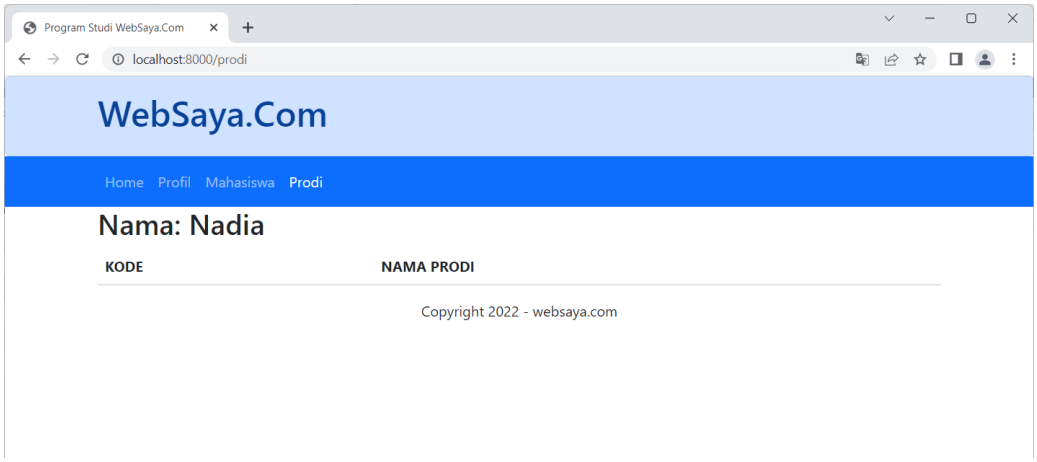
Untuk menampilkan data kateogri produk buatlah satu fungsi bernama index() pada prodiController.php dengan skript sebagai berikut:

```
public function index(){
    $title = "Program Studi WebSaya.Com";
    $slug = "prodi";
    $mhs = "Nadia";
    $dataProdi = prodi::all();
    return view('prodi.index',
            compact('mhs','title','slug','dataProdi'));
}
```

Fungsi index() diatas akan menampilkan view index.blade.php yang berada pada folder resources/views/prodi, karena view ini belum dibuat maka buatlah folder prodi terlebih dahulu, kemudian membuat file baru pada folder resources/views/prodi dengan nama index.blade.php dan isikan skript seperti dibawah:

```
@extends('layouts.main')
@section('title',$title)
@section('content')
    <h2>Nama: {{ $mhs}}</h2>
    <table class="table">
        <thead>
          <tr>
            <th scope="col">KODE</th>
            <th scope="col">NAMA PRODI</th>
          </tr>
        </thead>
        <tbody>
        @foreach ($dataProdi as $item)
        <tr>
            <td>{{ $item->kd_prodi }}</td>
            <td>{{ $item->nama_prodi }} </td>
          </tr>
        @endforeach
    </tbody>
</table>
@endsection
```

Sekarang jika kita mengakses link http://**localhost:8000/prodi** (pastikan server artisan sudah berjalan) maka halaman browser akan menampilkan halaman seperti berikut:



Pada Eloqunet ORM untuk melakukan operasi tambah, edit dan menghapus data tedapat dua cara yang pertama menggunakan Eloqunet ORM biasa atau menggunakan Eloquent ORM Mass Assignment. Perbedaan dari kedua cara ini terlihat pada penulisan syntaknya.

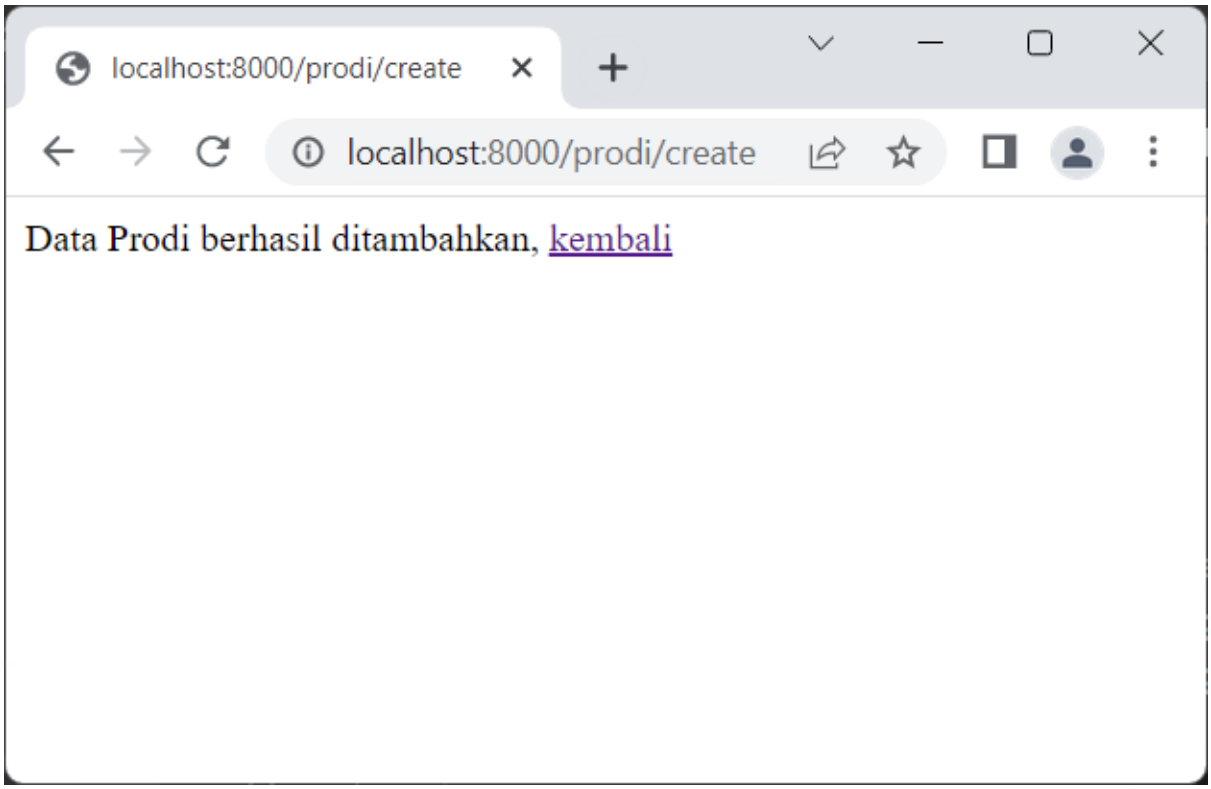
MENAMBAH DATA

Untuk menambahkan data menggunakan Eloquent ORM langkah-langkahnya sama dengan Query Builder. Pertama pada file prodiController.php buatlah satu fungsi baru bernama **create()** dan tambahkan koding dengan script berikut:

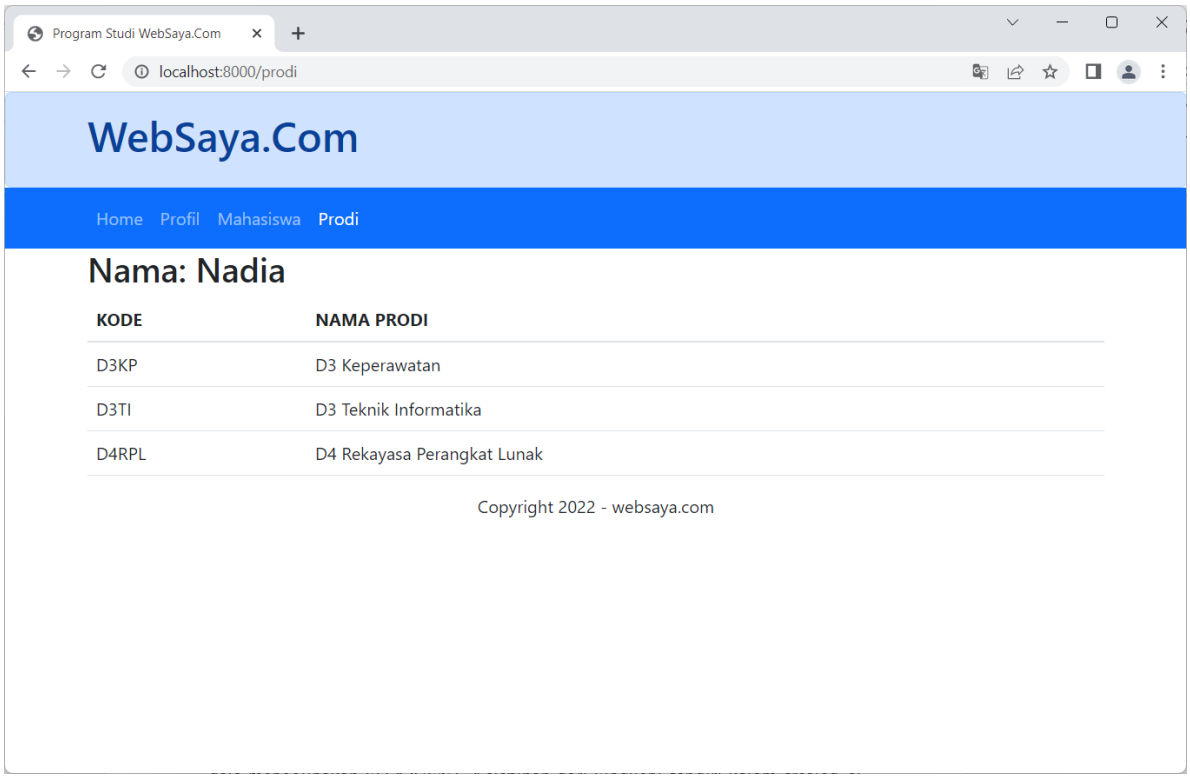
```
public function create()
{
    //Eloquent ORM biasa
    $dataProdi = new prodi();
    $dataProdi->kd_prodi = 'D3TI';
    $dataProdi->nama_prodi = 'D3 Teknik Informatika';
    $dataProdi->created_at = now();
    $dataProdi->updated_at = now();
    $dataProdi->save();

    //Eloquent ORM Mass Assignment
    prodi::insert(array(
                [
                    'kd_prodi' => 'D4RPL',
                    'nama_prodi' => 'D4 Rekayasa Perangkat Lunak',
                    'created_at' => now(),
                    'updated_at' => now()
                    ],
                [
                    'kd_prodi' => 'D3KP',
                    'nama_prodi' => 'D3 Keperawatan',
                    'created_at' => now(),
                    'updated_at' => now()
                    ],
    ));
    echo "Data Prodi berhasil ditambahkan,
        <a href='/prodi'>kembali</a>";
}
```

Kemudian cobalah jalankan route http://localhost:8000/prodi/create maka tampilanya akan menjadi seperti berikut:



Dan jika dilihat pada laman http://localhost:8000/prodi maka akan ada 3 data baru seperti pada gambar dibawah:



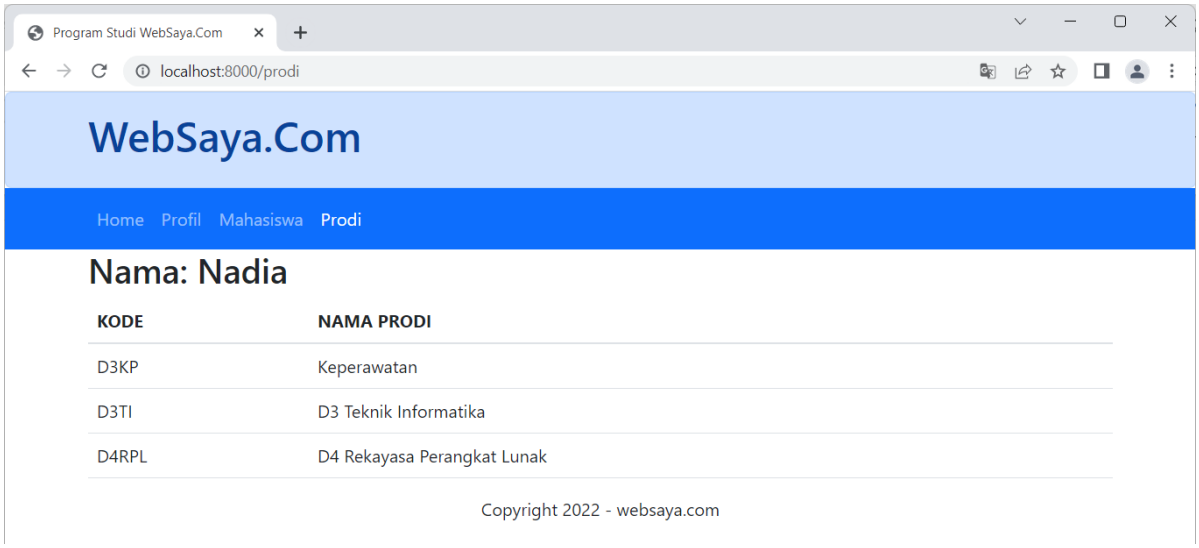
Dapat dilihat pada data diatas adalah hasil dari script menyimpan data menggunakan ELOQUENT. Kelebihan dari Eloquent sendiri kolom **created_at** dan **updated_at** akan terisi otomatis tergantung dari kapan data itu dibuat dan kapan data itu diperbaharui.

MEMPERBAHARUI DATA

Untuk memperbaharui data menggunakan ELOQUENT langkah-langkahnya sama dengan Query Builder yaitu pertama kita harus mendefinisikan data mana yang akan di update dan diikuti oleh nilai baru dari kolom. Pada folder **app\Http\Controller** buat satu fungsi baru bernama update() dan isikan script seperti berikut:

```
public function update()
{
    prodi::where('kd_prodi', '=', 'D3KP')->update([
        'nama_prodi' => 'Keperawatan'
    ]);
    echo "Data prodi D3KP berhasil di perbaharui,
        <a href='/prodi'>kembali</a>";
}
```

Pada script diatas adalah sintak untuk memperbaharui data menggunakan Eloquent Mass Assignment, dan sekarang jika kita menjalankan **route** http://localhost:8000/prodi/update maka hasilnya data kategori yang telah diperbaharui seperti pada gambar dibawah:

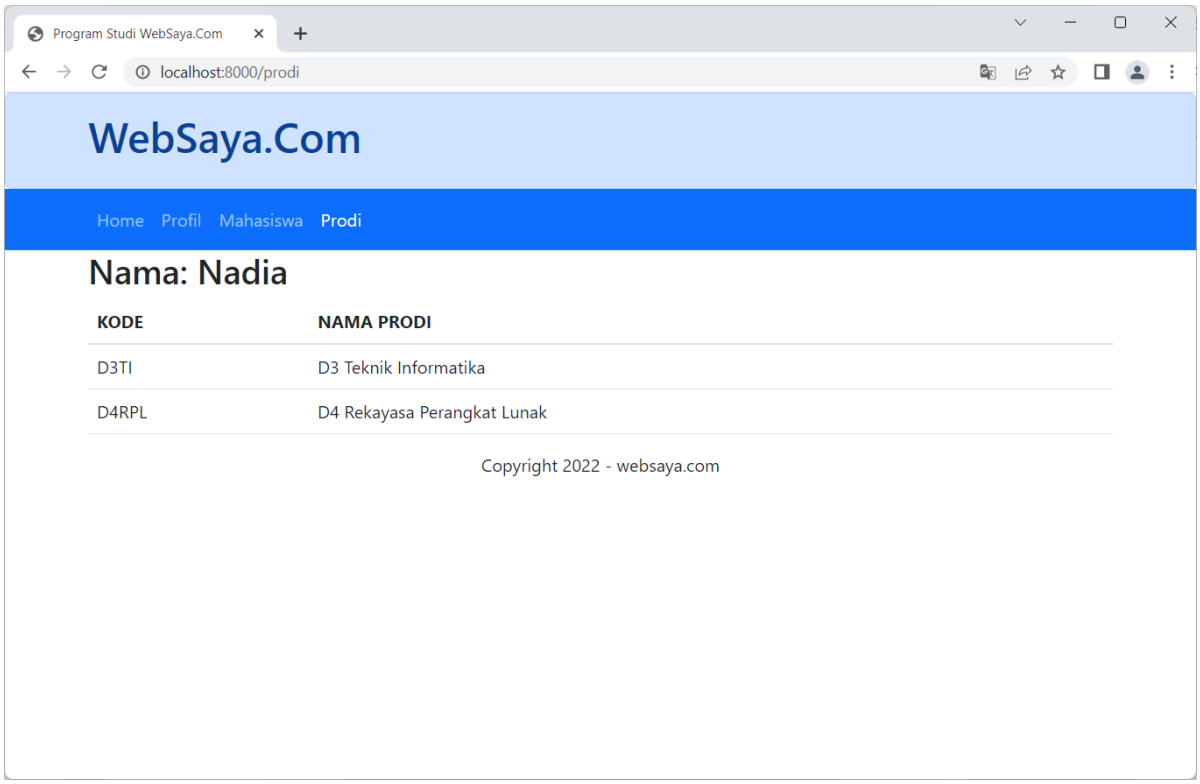


MENGHAPUS DATA

Untuk menghapus data menggunakan Eloquent kita harus mendefinisikan terlebih dahulu data mana yang akan dihapus dan diikuti dengan fungsi delete() untuk menghapus data tersebut dari tabel. buatlah satu fungsi baru bernama delete() pada file kateogriController.php dan isikan script seperti dibawah.

```php
public function destroy()
{
    prodi::where('kd_prodi','D3KP')->delete();
    return redirect('/prodi');
}
```

Kemudian jalankanlah route http://**localhost:8000/prodi/destroy** untuk mengesekusi method tersebut maka kita akan diarahkan ke kembali ke route http://**localhost:8000/prodi** dengan data prodi yang mempunyai kd_prodi D3KP yang telah dihapus seperti pada gambar dibawah:



TUGAS

1. Buatlah sebuah halaman mahasiswa untuk melakukan proses tampil, tambah data, perubahan data, dan hapus.
2. Proses pembuatan pada poin 1 menggunakan ELOQUENT ORM
3. Lakukan prosesnya seperti pada praktikum tabel prodi
4. Layout halamannya menggunakan master template blade

Catatan:

- Langkah-langkah pembuatannya dapat mengikuti praktikum
- Laporan praktikum tugas dan sourcecode dikumpulkan ke elearning