



Eloquent ORM

Definisi

- Cara pengaksesan database dimana setiap baris tabel dianggap sebagai sebuah object
- **ORM** sendiri merupakan singkatan dari Object-relational mapping → sebuah teknik programming untuk mengkonversi data ke dalam bentuk object
- database terdiri dari kumpulan tabel yang saling terhubung, data disimpan dalam bentuk baris dan kolom
- ORM dipakai untuk mengubah baris dan kolom ini menjadi sebuah object
- setiap kolom akan menjadi property dari object tersebut
- Kode program menjadi lebih rapi dan 'seragam' dengan fitur OOP lain di Laravel
- **Eloquent** adalah "nama" dari implementasi ORM yang digunakan Laravel
- Doctrine ORM yang dipakai framework Symfony

Persiapan

- Jalankan perintah migration untuk mengosongkan database

`php artisan migrate:fresh`

```
Dropping all tables ..... 124ms DONE
```

```
INFO Preparing database.
```

```
Creating migration table ..... 80ms DONE
```

```
INFO Running migrations.
```

```
2014_10_12_000000_create_users_table ..... 60ms DONE
```

```
2014_10_12_100000_create_password_resets_table ..... 28ms DONE
```

```
2019_08_19_000000_create_failed_jobs_table ..... 34ms DONE
```

```
2019_12_14_000001_create_personal_access_tokens_table ..... 44ms DONE
```

```
2022_11_13_180437_create_mahasiswa_table ..... 38ms DONE
```

- Copy MahasiswaController, sebagai backup, Hapus semua function dan sisakan kode seperti contoh

MahasiswaController

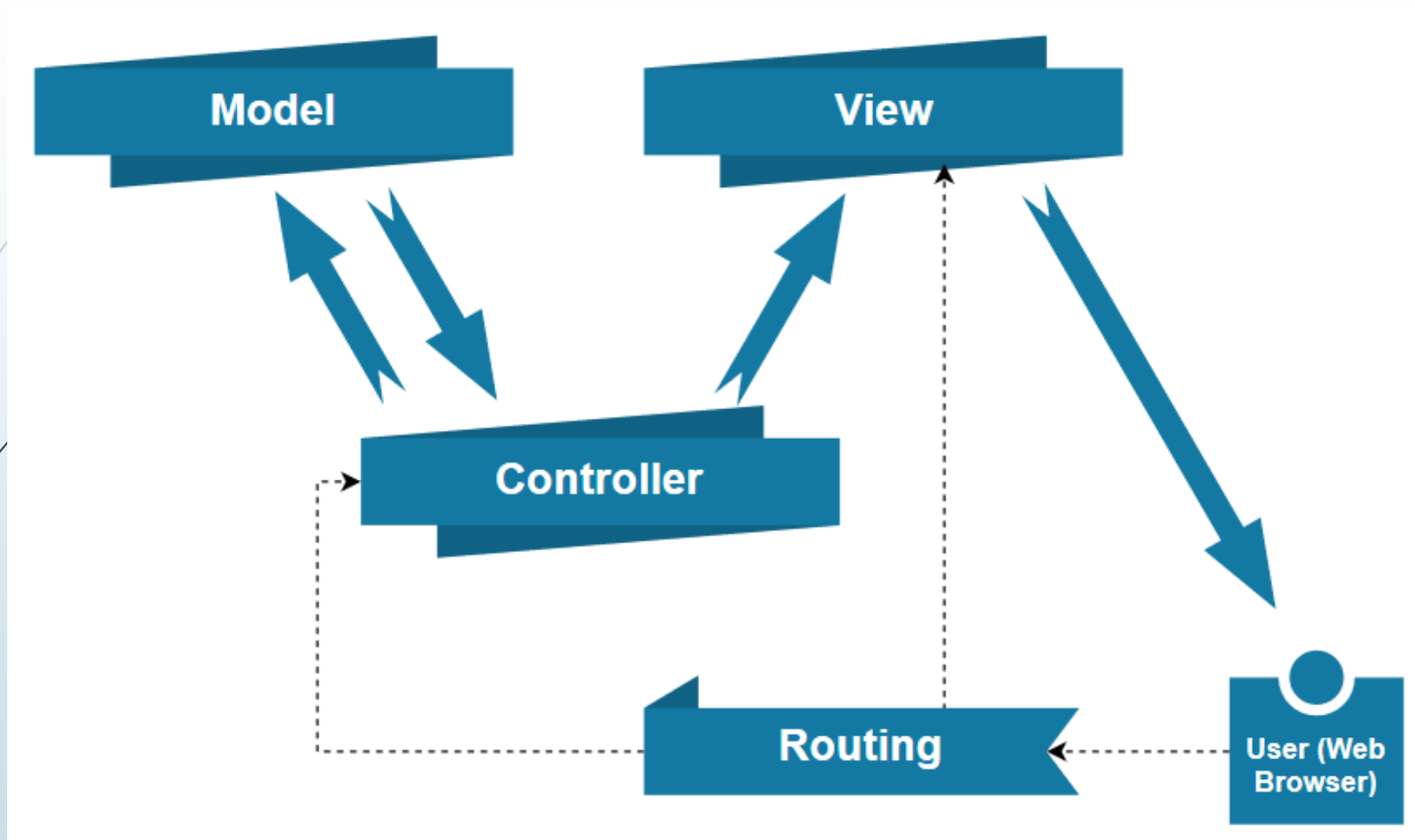
```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class MahasiswaController extends Controller
8  {
9      //Tempat menuliskan kode
10 }
```

- Tidak membutuhkan class DB

Route

```
41 Route::get('/cek-object', [MahasiswaController::class, 'cekObject']);
42 Route::get('/insert', [MahasiswaController::class, 'insert']);
43 Route::get('/mass-assignment', [MahasiswaController::class, 'massAssignment']);
44 Route::get('/mass-assignment2', [MahasiswaController::class, 'massAssignment2']);
45 Route::get('/update', [MahasiswaController::class, 'update']);
46 Route::get('/update-where', [MahasiswaController::class, 'updateWhere']);
47 Route::get('/mass-update', [MahasiswaController::class, 'massUpdate']);
48 Route::get('/delete', [MahasiswaController::class, 'delete']);
49 Route::get('/destroy', [MahasiswaController::class, 'destroy']);
50 Route::get('/mass-delete', [MahasiswaController::class, 'massDelete']);
51 Route::get('/all', [MahasiswaController::class, 'all']);
52 Route::get('/all-view', [MahasiswaController::class, 'allView']);
53 Route::get('/get-where', [MahasiswaController::class, 'getWhere']);
54 Route::get('/test-where', [MahasiswaController::class, 'testWhere']);
55 Route::get('/first', [MahasiswaController::class, 'first']);
56 Route::get('/find', [MahasiswaController::class, 'find']);
57 Route::get('/latest', [MahasiswaController::class, 'latest']);
58 Route::get('/limit', [MahasiswaController::class, 'limit']);
59 Route::get('/skip-take', [MahasiswaController::class, 'skipTake']);
60 Route::get('/soft-delete', [MahasiswaController::class, 'softDelete']);
61 Route::get('/with-trashed', [MahasiswaController::class, 'withTrashed']);
62 Route::get('/restore', [MahasiswaController::class, 'restore']);
63 Route::get('/force-delete', [MahasiswaController::class, 'forceDelete']);
```

MVC



Model

- Eloquent ORM memerlukan Model untuk proses konversi data tabel menjadi object
- Perintah artisan

```
php artisan make:model <namaModel>
```

- Nama model berpasangan dengan nama table, namun dalam bentuk singular
- Contoh nama table mahasiswa ➔ nama model Mahasiswa
- Jalankan perintah berikut di terminal

```
php artisan make:model Mahasiswa
```


Model Mahasiswa.php

app > Models >  Mahasiswa.php > ...


```
1  k?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Mahasiswa extends Model
9  {
10     use HasFactory;
11 }
12
```

Cek Object

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\Mahasiswa;
7
8  class MahasiswaController extends Controller
9  {
10     public function cekObject(){
11         $mahasiswa = new Mahasiswa;
12         dump($mahasiswa);
13     }
14 }
```

- Tambahkan baris ke 6, dan 10 – 13 pada Mahasiswa Controller
- Akses ke alamat localhost/cek-object

Hasil cek-object



```
← → ↻ 127.0.0.1:8000/cek-object

App\Models\Mahasiswa {#295 ▾ // app\Http\Controllers\MahasiswaController.php:12
  #connection: null
  #table: null
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: false
  +wasRecentlyCreated: false
  #escapeWhenCastingToString: false
  #attributes: []
  #original: []
  #changes: []
  #casts: []
  #classCastCache: []
  #attributeCastCache: []
  #dates: []
  #dateFormat: null
  #appends: []
  #dispatchesEvents: []
  #observables: []
  #relations: []
  #touches: []
  +timestamps: true
  #hidden: []
  #visible: []
  #fillable: []
  #guarded: array:1 [▶]
}
```

Input Data

```
15 public function insert(){
16     $mahasiswa = new Mahasiswa;
17     $mahasiswa->nim = '320201601';
18     $mahasiswa->nama = 'Armand Maulana';
19     $mahasiswa->tanggal_lahir = '2001-12-31';
20     $mahasiswa->ipk = 3.5;
21     $mahasiswa->save();
22     dump($mahasiswa);
23 }
```

- Akses ke localhost/insert
- Di dalam bagian #attributes bisa terlihat semua property yang baru saja kita input, ditambah kolom khusus yang di-generate secara otomatis oleh Laravel, seperti kolom id, serta kolom updated_at dan kolom created_at yang juga otomatis berisi data timestamp

```
#escapeWhenCastingToString: false
#attributes: array:7 [▼
    "nim" => "320201601"
    "nama" => "Armand Maulana"
    "tanggal_lahir" => "2001-12-31"
    "ipk" => 3.5
    "updated_at" => "2022-11-13 22:41:29"
    "created_at" => "2022-11-13 22:41:29"
    "id" => 1
]
#original: array:7 [▶]
```

Mass Assignment

- Bisa mengisi banyak property untuk object Mahasiswa dalam sekali proses.
- Cara input mass assignment dilakukan dengan mengakses static method create() dari Model object → berguna untuk pemrosesan form → Bawaan Laravel associative array

```
25     public function massAssignment(){
26         Mahasiswa::create(
27             [
28                 'nim'           => '320201602',
29                 'nama'          => 'Ruth Sahanaya',
30                 'tanggal_lahir' => '2003-08-22',
31                 'ipk'           => 3.5,
32             ]
33         );
34         return "Berhasil di proses";
35     }
```

Mass assignment



- Akses ke alaman localhost/mass-assignment
- Muncul error karena Laravel membatasi akses table ketika diproses menggunakan mass assignment
- Laravel mewajibkan programmer untuk menulis nama kolom apa saja yang boleh diinput
- Sehingga perlu mendaftarkan kolom 'nim', 'nama', 'tanggal_lahir' dan 'ipk' ke dalam property **\$fillable** yang ada di model **Mahasiswa**

Mass Assignment

- ➡ Buka file Model Mahasiswa
- ➡ Tambahkan kode pada baris ke-11

```
8  class Mahasiswa extends Model
9  {
10     use HasFactory;
11     protected $fillable = ['nim', 'nama', 'tanggal_lahir', 'ipk'];
12 }
```

- ➡ Akses kembali localhost/mass-assignment

Property \$guarded

- \$guarded dipakai untuk menulis nama kolom apa saja yang tidak boleh diisi
- Contoh jika kolom ipk tidak boleh diisi menggunakan mass assignment

```
protected $guarded = ['ipk'];
```

- Jika tidak ada yang dibatasi, tuliskan array kosong

```
protected $guarded = [];
```


Mass Assignment 2

- Akses ke alamat localhost/mass-assignment2

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController.php

37 public function massAssignment2()
38 {
39     $mahasiswa1 = Mahasiswa::create(
40         [
41             'nim'          => '320201603',
42             'nama'         => 'Titi DJ',
43             'tanggal_lahir' => '2003-10-19',
44             'ipk'          => '3.01',
45         ]
46     );
47     dump($mahasiswa1);
48
49     $mahasiswa2 = Mahasiswa::create(
50         [
51             'nim'          => '320201604',
52             'nama'         => 'Kaka',
53             'tanggal_lahir' => '2005-04-09',
54             'ipk'          => '3.33',
55         ]
56     );
57     dump($mahasiswa2);
58
59     $mahasiswa3 = Mahasiswa::create(
60         [
61             'nim'          => '320201605',
62             'nama'         => 'Ario Wahab',
63             'tanggal_lahir' => '2003-01-01',
64             'ipk'          => '3.53',
65         ]
66     );
67     dump($mahasiswa3);
68 }
```

Update Data (find)

- Eloquent ORM memproses data tabel menggunakan object.
- Caranya → cari Model object dari tabel, lalu ubah beberapa property dan simpan
- Akses ke alamat localhost/update

```
70     public function update(){
71         $mahasiswa = Mahasiswa::find(1);
72         $mahasiswa->tanggal_lahir = '2001-12-31';
73         $mahasiswa->ipk = 3.50;
74         $mahasiswa->save();
75         dump($mahasiswa);
76     }
77
```

Update data

- ➡ Method `find()` dipakai untuk mencari data tabel berdasarkan kolom id. Artinya, perintah `Mahasiswa::find(1)` akan mengambil data di tabel mahasiswa yang memiliki id = 1, lalu membuat object dari data tersebut.
- ➡ Object ini selanjutnya disimpan ke dalam variabel `$mahasiswa`
- ➡ Baris 72 dan 73 mengisi data baru
- ➡ Agar perubahan tersimpan ke database, akhiri dengan perintah `$mahasiswa->save()`

Update Data

- Dari hasil dump(\$mahasiswa), silahkan buka tab #changes, di dalamnya berisi 2 data yang berubah, yakni ipk, dan update_at

```
#escapeWhenCastingToString: false
#attributes: array:7 [▶]
#original: array:7 [▶]
#changes: array:2 [▼]
    "ipk" => 3.5
    "updated_at" => "2022-11-14 01:35:05"
]
#casts: []
```

Update data (where)

```
78 public function updateWhere(){
79     $mahasiswa = Mahasiswa::where('nim', '320201605')->first();
80     $mahasiswa->tanggal_lahir = '2010-12-31';
81     $mahasiswa->ipk = 4.0;
82     $mahasiswa->save();
83     dump($mahasiswa);
84 }
```

➡ Akses ke alamat localhost/update-where

```
    "nama" => "Ario Wahab"
    "tanggal_lahir" => "2010-12-31"
    "ipk" => 4.0
    "created_at" => "2022-11-14 01:26:43"
    "updated_at" => "2022-11-14 03:46:40"
]
#original: array:7 [▼
    "id" => 5
    "nim" => "320201605"
    "nama" => "Ario Wahab"
    "tanggal_lahir" => "2010-12-31"
    "ipk" => 4.0
    "created_at" => "2022-11-14 01:26:43"
    "updated_at" => "2022-11-14 03:46:40"
]
#changes: array:3 [▼
    "tanggal_lahir" => "2010-12-31"
    "ipk" => 4.0
    "updated_at" => "2022-11-14 03:46:40"
]
#casts: []
```

Menghapus Data (delete)

```
86     public function delete(){
87         $mahasiswa = Mahasiswa::find(1);
88         $mahasiswa->delete();
89         dump($mahasiswa);
90     }
```

- Akses ke alamat localhost/delete
- Coba eksekusi lebih dari 1 kali, apa yang terjadi

```
#escapeWhenCastingToString: false
#attributes: array:7 [▼
    "id" => 1
    "nim" => "320201601"
    "nama" => "Armand Maulana"
    "tanggal_lahir" => "2001-12-31"
    "ipk" => "3.50"
    "created_at" => "2022-11-13 22:41:29"
    "updated_at" => "2022-11-14 01:35:05"
]
```

Menghapus Data (destroy)

```
public function destroy(){  
    $mahasiswa = Mahasiswa::destroy(1);  
    dump($mahasiswa);  
}
```

- ➡ Akses ke alamat localhost/delete
- ➡ Coba eksekusi lebih dari 1 kali, apa yang terjadi ?

```
#escapeWhenCastingToString: false  
#attributes: array:7 [▼  
    "id" => 1  
    "nim" => "320201601"  
    "nama" => "Armand Maulana"  
    "tanggal_lahir" => "2001-12-31"  
    "ipk" => "3.50"  
    "created_at" => "2022-11-13 22:41:29"  
    "updated_at" => "2022-11-14 01:35:05"  
]
```

Menampilkan Data (all)

```
public function all(){  
    $result = Mahasiswa::all();  
    dump($result);  
}
```

- ➡ Akses ke alamat localhost/all

Menampilkan data pertama

```
public function all(){  
    $result = Mahasiswa::all();  
    echo($result[0]->id). '<br>';  
    echo($result[0]->nim). '<br>';  
    echo($result[0]->nama). '<br>';  
    echo($result[0]->tanggal_lahir). '<br>';  
    echo($result[0]->ipk);  
}
```

- ➡ Akses ke alamat localhost/all

Menampilkan semua data

```
public function all(){  
    $result = Mahasiswa::all();  
    foreach ($result as $mahasiswa) {  
        echo($mahasiswa->id). '<br>';  
        echo($mahasiswa->nim). '<br>';  
        echo($mahasiswa->nama). '<br>';  
        echo($mahasiswa->tanggal_lahir). '<br>';  
        echo($mahasiswa->ipk). '<br>';  
        echo "<hr>";  
    }  
}
```

➡ Akses ke alamat localhost/all

2
320201602
Ruth Sahanaya
2003-08-22
3.50

3
320201603
Titi DJ
2003-10-19
3.01

4
320201604
Kaka
2005-04-09
3.33

5
320201605
Ario Wahab
2010-12-31
4.00

Mengirim data ke View

```
109 public function allView(){
110     $mahasiswas = Mahasiswa::all();
111     return view('tampil-mahasiswa', ['mahasiswas' => $mahasiswas]);
112 }
```

➡ Akses ke alamat localhost/all-view

Method where

```
114 public function getWhere(){
115     $mahasiswas = Mahasiswa::where('ipk','>','3.5')
116         ->orderBy('nama', 'desc')
117         ->get();
118     return view('tampil-mahasiswa',['mahasiswas' => $mahasiswas]);
119 }
```

- ➡ Untuk mengambil sebagian data
- ➡ Akses ke alamat localhost/get-where

Method first

```
121  ✓ public function first(){  
122      $mahasiswa = Mahasiswa::where('nim','320201605')->first();  
123      return view('tampil-mahasiswa',['mahasiswas' => [$mahasiswa]]);  
124  }
```

- ➡ Untuk mengambil elemen pertama dari hasil
- ➡ Akses ke alamat localhost/first

Soft delete

Eloquent tidak benar-benar menghapus data dari dalam tabel, tapi menambah satu kolom khusus sebagai penanda data yang telah dihapus.

- Dengan method `all()`, data ini tidak akan terlihat
- Syarat → memiliki kolom `delete_at`, import class `Illuminate\Database\Eloquent\SoftDeletes` ke dalam Model
- Tambahkan baris 23 pada file migration

```
14     public function up()
15     {
16         Schema::create('mahasiswas', function (Blueprint $table) {
17             $table->id();
18             $table->char('nim',10)->unique();
19             $table->string('nama');
20             $table->date('tanggal_lahir');
21             $table->decimal('ipk',3,2)->default(1.00);
22             $table->timestamps();
23             $table->softDeletes();
24         });
25     }
```

Soft delete

- Jalankan perintah :
php artisan migrate:fresh
*akan menyebabkan isi table sebelumnya hilang
- Jalankan perintah:
localhost:8000/insert
- Jalankan perintah
localhost:8000/mass-assignment
- Jalankan perintah:
localhost:8000/mass-assignment2
- Cek data (terdapat 5 record), dengan satu kolom tambahan

Mahasiswa Model

```
app > Models > Mahasiswa.php > Mahasiswa
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\SoftDeletes;
8
9  class Mahasiswa extends Model
10 {
11     use HasFactory;
12     protected $fillable = ['nim', 'nama', 'tanggal_lahir', 'ipk'];
13     use SoftDeletes;
14 }
15
```

➡ Tambahkan baris 7 dan 13 pada mahasiswa model

Percobaan hapus

```
126     public function softDelete()  
127     {  
128         Mahasiswa::where('nim', '320201605')->delete();  
129         return "Berhasil di hapus";  
130     }
```

- Akses ke alamat localhost/soft-delete
- Kemudian cek database
- Kemudian akses ke alamat localhost/all-view
- Apa yang terjadi ?
- Bagaimana jika diakses dengan raw query atau query builder ?

Menampilkan semua data, termasuk softdelete

```
132 public function withTrashed(){  
133     $mahasiswas = Mahasiswa::withTrashed()->get();  
134     return view('tampil-mahasiswa',['mahasiswas' => $mahasiswas]);  
135 }
```

- ➡ Akses ke alamat localhost/with-trashed

Restore soft delete

```
137 public function restore(){  
138     Mahasiswa::withTrashed()->where('nim','320201605')->restore();  
139     return "Berhasil di restore";  
140 }
```

➡ Akses ke alamat localhost/restore

Hapus Data Permanen

```
142 public function forceDelete(){  
143     Mahasiswa::where('nim','320201605')->forceDelete();  
144     return "Berhasil di hapus secara permanen";  
145 }
```

- ➡ Akses ke alamat localhost/force-delete