Eloquent ORM - Laravelin tietokannan käsittely

Laravelin relaatiotietokantakäsittely perustuu oletusarvoisesti Eloquent ORMiin. ORMia käyttäen tietokannan sisältöä voidaan käsitellä olioiden avulla. Laravelissa tietokannan taulut luodaan tyypillisesti `migrations`-luokkia käyttämällä `artisan`-skriptillä, jonka voi ajatella olevan eräänlainen versionhallintatyökalu tauluille. Luodusta tietokantarakenteesta voidaan esim. peruuttaa luontia edeltäneeseen tilaan.

Tässä luvussa oletetaan, että olet asentanut järjestelmääsi MySQL-tietokantapalvelimen käyttöönottoluvussa esitetyllä tavalla. Esityksessä myös oletaan ensimmäisiä esimerkkejä lukuunottamatta, että olet asiayhteyden perusteella tietoinen milloin esitetyt komennot annetaan Linuxin komentorviltä, milloin mysql-kehotteeseen ja milloin ollaan esim. editoimassa jonkin tiedoston sisältöä. Muistutetaan lisäksi, että kansioviittaukset ovat suhteessa Laravelin projektikansioon `~/public_html/projekti01` ja esim. `artisan`-komennot on annettava ao. kansiossa.

27.01 Tietokannan yhteystiedot

Tietokannan yhteystiedot määritellään Laravel-projektin juurihakemiston `.env`-tiedostossa. Tietokannan nimenä valitaan käytettäväksi `laraveldb`. Tiedostoa `.env` voi editoida vaikkapa Linux-komentoriviltä `nano`-editorilla

```
nano .env
```

Editoidaaan .env-tiedostooon seuraavat arvot

.env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laraveldb
```

```
DB_USERNAME=root
DB_PASSWORD=sala
```

27.02 Tietokannan luominen

Luodaan `.env`-tiedostossa määritelty tietokanta Linux-komentoriviltä. Otetaan yhteys tietokantapalvelimeen

```
mysql -u root -psala
```

ja luodaan `laraveldb`-niminen tietokanta:

```
mysql> create database laraveldb;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| laraveldb          |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
6 rows in set (0.02 sec)

mysql> exit
```

27.03 Tutkitaan Laravelin valmiita tietokantatauluja

Perusasennetun Laravelin mukana on määritelty automaattisesti luotavaksi `users`-, `password_resets`- ja `failed_jobs`-taulut mm. käyttäjätietojen tallentamista ja tunnistautumismekanismeja (autentikointi) varten. Näiden kuten kaikkien muidenkin itse luotavien tietokantamigraatioita hyödyntävien taulujen määritykset on esitetty Laravelin projektikansion `database/migrations`-kansion `migrations`-luokissa php-skripteillä.

```
ls

2014_10_12_000000_create_users_table.php
2014_10_12_100000_create_password_resets_table.php
2019_08_19_000000_create_failed_jobs_table.php
```

Enempiä selittelemättä huomataan, että `2014_10_12_000000_create_users_table.php`-tiedostossa on määritelty `users`-taulun tietokannan kentät tyyppeineen

```
public function up()
{
        Schema::create('users', function (Blueprint $table) {
                $table->bigIncrements('id');
                $table->string('name');
                $table->string('email')->unique();
                $table->timestamp('email_verified_at')->nullab
                $table->string('password');
                $table->rememberToken();
                $table->timestamps();
        });
}
```

Ajetaan määritellyt migraatiot tietokantaan komennolla

```
php artisan migrate
```

komennon tuloksena näkyy

```
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.11 sec
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.
```

Tutkitaan tulosta varsinaisesta tietokannasta

```
mysql -u root -psala laraveldb
```

Huomataan syntyneet taulut kenttineen ja todetaan että sisältö on tyhjä:

```
mysql> show tables;
+----------------------+
| Tables_in_laraveldb |
+----------------------+
| failed_jobs          |
| migrations           |
| password_resets      |
| users                |
+----------------------+
4 rows in set (0.00 sec)

mysql> desc users;
+--------------------+----------------------+------+-----+-
| Field              | Type                 | Null | Key | D
+--------------------+----------------------+------+-----+-
| id                 | bigint(20) unsigned  | NO   | PRI | N
| name               | varchar(255)         | NO   |     | N
| email              | varchar(255)         | NO   | UNI | N
| email_verified_at  | timestamp            | YES  |     | N
| password           | varchar(255)         | NO   |     | N
| remember_token     | varchar(100)         | YES  |     | N
| created_at         | timestamp            | YES  |     | N
| updated_at         | timestamp            | YES  |     | N
+--------------------+----------------------+------+-----+-
8 rows in set (0.01 sec)

mysql> select * from users;
Empty set (0.00 sec)
```

Migraatiomekanismiin kuuluu mahdollisuus myös peruuttaa tehdyt migraatiot. Näin on mahdollista muokata tietokannan syntyvää rakennetta migraatiotiedostojen avulla: Peruutetaan (rollback) luodut migraatiot, muokataan tiedostoja ja ajetaan migraatiot uudelleen tietokantaan.

Peruutetaan edellä luodut migraatiot:

```
php artisan migrate:rollback
```

ja todetaan taulujen poistaminen

```
mysql> show tables;
+----------------------+
| Tables_in_laraveldb |
+----------------------+
| migrations           |
+----------------------+
1 row in set (0.00 sec)
```

27.04 Luodaan oma customers-taulu

Luodaan lähes identtinen tietokannan taulu aiemmin luvussa "*MySQL-esimerkkitietokannan luominen*" esitetyn customer-taulun kanssa. Luodaan migraatiotiedosto

```
php artisan make:migration create_customers_table
Created Migration: 2019_10_05_051206_create_customers_tab
```

ja avataan se `database_migrations`-kansiosta sisältö näyttää kokonaisuudessaan seuraavalta

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7
8  class CreateCustomersTable extends Migration
9  {
10     /**
11      * Run the migrations.
12      *
13      * @return void
14      */
15     public function up()
16     {
17         Schema::create('customers', function (Blueprint $tabl
```

```
            $table->bigIncrements('id');
            $table->timestamps();
        });
    }

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

Muutetaan `up()`-metodia tarpeita vastaavaksi lisäämällä määrittelyyn `id`- ja `birth_date`-kentät:

```
public function up()
{
        Schema::create('customers', function (Blueprint $tabl
                $table->bigIncrements('id');
                $table->string('name');
                $table->date('birth_date');
                $table->timestamps();
        });
}
```

Ajetaan migraatiot. Huomaa: Jos peruutit aiemmin tehdyn migraation, niin nyt luodaan uudelleen customers-taulun lisäksi myös aiemmat 3 taulua

```
php artisan migrate
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.14 sec
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.
Migrating: 2019_10_05_051206_create_customers_table
Migrated:  2019_10_05_051206_create_customers_table (0.06
```

`desc customers` -komennon tuoksena todetaan, että PHP-merkinnällä `$table->bigIncrements('id');` luodaan `id`-niminen autoincrement-asetuksen mukainen kenttä ja PHP-merkinnällä `$table->timestamps();` luodaan kaksi timestamp-kenttää `created_at` ja `updated_at`. Laravelin käyttämä Eloquent ORM olettaa kaikilta osin toimiakseen, että tietokannan taulussa on nämä kolme kenttää `id`, `created_at` ja `updated_at` Ota tämä jatkossa huomioon.

```
mysql> desc customers;
+------------+---------------------+------+-----+---------+
| Field      | Type                | Null | Key | Default |
+------------+---------------------+------+-----+---------+
| id         | bigint(20) unsigned | NO   | PRI | NULL    |
| name       | varchar(255)        | NO   |     | NULL    |
| birth_date | date                | NO   |     | NULL    |
| created_at | timestamp           | YES  |     | NULL    |
| updated_at | timestamp           | YES  |     | NULL    |
+------------+---------------------+------+-----+---------+
5 rows in set (0.00 sec)
```

Tutustutaan Eloquent ORMiin tinkerin avulla

27.05 Mallin luominen

Luodaan luokka `Customer`, joka toimii **mallina** `customers`-taulun käsittelylle. Laravelissa taulut nimetään monikossa esim. `customers` (pieni alkukirjain). Vastaava taulua käsittelevä malli esittää yhtä tauluun tallennettavaa tietuetta ja se nimetään yksikössä esim. `Customer` (iso alkukirjain).

Malli luodaan artisan-komennolla

```
php artisan make:model Customer
```

Tuloksena syntyy tiedosto `App/Customer.php`

```php
<?php
namespace App;
```

```
3  use Illuminate\Database\Eloquent\Model;
4  class Customer extends Model
5  {
6      //
7  }
```

joka perii Model-luokasta kaiken kaiken keskeisen tietokannan käsittelyä tarvittavan toiminnallisuuden.

27.06 Eloquent - Tinker

Tutustutaan Eloquent ORMiin aluksi `tinker`-komentorivikäyttöliittymän avulla. Käynnistetään tinker `php artisan tinker`-komennolla. Kehotteena tinkerissä toimii `>>>`

```
php artisan tinker
Psy Shell v0.9.9 (PHP 7.2.22-1+ubuntu18.04.1+deb.sury.org
>>>
```

Luodaan mallin avulla uusi $customer-olio ja asetetaan arvot kahdelle itse määritellylle kentälle

```
>>> $customer  = new App\Customer;
=> App\Customer {#3011}
>>> $customer->name = "Guru Ken";
=> "Guru Ken"
>>> $customer->birth_date = "2011-11-11";
=> "2011-11-11"
>>> $customer;
=> App\Customer {#3011
     name: "Guru Ken",
     birth_date: "2011-11-11",
   }
```

Tässä vaiheessa voit halutetessasi tarkistaa esim. toisesta auki olevasta pääteikkunasta, että tietuetta ei ole vielä tallennettu tietokantaan.

```
mysql -u root -psala laraveldb
mysql> select * FROM customers;
```

```
Empty set (0.00 sec)
```

mutta kirjoittamalla tinkerissä

```
>>> $customer->save();
=> true
```

sisältö tallentuu tietokantaan. Keskeisesti mm. tätä on ORM (Object-relational Mapping). Olion ominaisuuksina saatavilla oleva data tallennetaan relaatiotietokannan tiettueeksi.

```
mysql> select * FROM customers;
+----+----------+------------+---------------------+-----
| id | name     | birth_date | created_at          | upda
+----+----------+------------+---------------------+-----
|  1 | Guru Ken | 2011-11-11 | 2019-10-05 08:15:00 | 2019
+----+----------+------------+---------------------+-----
1 row in set (0.00 sec)
```

Huom: Metodi `save()` tarvitsee toimiakseen `id`, `created_at` ja `updated_at`-nimiset kentät.

27.07 Eloquent - pieni esittelykierros

Uuden asiakkaan luominen - create()

```
>>> $customer = App\Customer::create(['name' => 'Alainen
Illuminate/Database/Eloquent/MassAssignmentException with
```

=> Ei onnistu. Pitää sallia kenttien **mass assignment** eli lisätä tiedostoon `app/Customer.php` eli Customer-luokan malliin seuraavaa `$fillable`-taulukko:

```
1  class Customer extends Model
2  {
```

```
    protected $fillable = [
        'name',
        'birth_date'
    ];
}
```

Tinker-pitää käynnistää uudelleen, että asetus on voimassa eli toimitaan seuraavasti

```
>>> exit
Exit:  Goodbye
testi@userver180401:~/public_html/projekti01$ php artisan
Psy Shell v0.9.9 (PHP 7.2.22-1+ubuntu18.04.1+deb.sury.org
>>> $customer = App\Customer::create(['name' => 'Alainen K
=> App\Customer {#3006
     name: "Alainen Kim",
     birth_date: "2012-12-12",
     updated_at: "2019-10-05 13:04:40",
     created_at: "2019-10-05 13:04:40",
     id: 2,
   }
>>>
```

Huomataan, että massatäytön onnistumisen lisäksi `create()`-metodia käytettäessä data tallennetaan samalla myös tietokantaan:

```
mysql> select * FROM customers;
+----+-------------+------------+---------------------+--
| id | name        | birth_date | created_at          | up
+----+-------------+------------+---------------------+--
|  1 | Guru Ken    | 2011-11-11 | 2019-10-05 08:15:00 | 20
|  2 | Alainen Kim | 2012-12-12 | 2019-10-05 13:04:40 | 20
+----+-------------+------------+---------------------+--
2 rows in set (0.00 sec)
```

Tietojen päivitys

Tapa 1: Luetaan tietue olio-muuttujaan, muutetaan halutut tiedot ja kutsutaan `save()`-metodia

```
>>> $customer = App\Customer::find(2);
=> App\Customer {#3033
```

```
        id: 2,
        name: "Alainen Kim",
        birth_date: "2012-12-12",
        created_at: "2019-10-05 13:04:40",
        updated_at: "2019-10-05 13:04:40",
      }
>>> $customer->name = "Ainen Sani";
=> "Ainen Sani"
>>> $customer->save();
=> true
```

Tarkastetaan tulos

```
mysql> select * FROM customers;
+----+------------+------------+---------------------+---
| id | name       | birth_date | created_at          | up
+----+------------+------------+---------------------+---
|  1 | Guru Ken   | 2011-11-11 | 2019-10-05 08:15:00 | 20
|  2 | Ainen Sani | 2012-12-12 | 2019-10-05 13:04:40 | 20
+----+------------+------------+---------------------+---
2 rows in set (0.00 sec)
```

Tapa 2: Luetaan tietue olio-muuttujaan ja kutsutaan `update()`-metodia

```
>>> $customer = App\Customer::find(2);
>>> ...
>>> $customer->update(['name' => 'Vainio Vilja']);
=> true
```

ja tarkistetaan tulos

```
mysql> select * FROM customers;
+----+--------------+------------+---------------------+-
| id | name         | birth_date | created_at          | u
+----+--------------+------------+---------------------+-
|  1 | Guru Ken     | 2011-11-11 | 2019-10-05 08:15:00 | 2
|  2 | Vainio Vilja | 2012-12-12 | 2019-10-05 13:04:40 | 2
+----+--------------+------------+---------------------+-
2 rows in set (0.00 sec)
```

Poistaminen - delete()

Tietueen poistaminen tapahtuu delete()-metodilla. Luodaan "Päivä Perho" ja poistetaan se

```
>>> $customer = App\Customer::create(['name' => 'Perho Pä
=> App\Customer {#3025
     name: "Perho Päivä",
     birth_date: "2019-10-05",
     updated_at: "2019-10-05 20:02:53",
     created_at: "2019-10-05 20:02:53",
     id: 3,
   }
>>> $customer->delete();
=> true
>>> App\Customer::all()->map->name;
=> Illuminate\Support\Collection {#3036
     all: [
       "Guru Ken",
       "Vainio Vilja",
     ],
   }
```

Tietokannan selaamista

Tietokannan sisältöä voi selata useilla erilaisilla Model-luokasta perittyjen metodien avulla. Tässä muutamia esimerkkejä.

Koko taulun sisältö kokoelmana

```
>>> App\Customer::all();
=> Illuminate\Database\Eloquent\Collection {#3015
     all: [
       App\Customer {#3007
         id: 1,
         name: "Guru Ken",
         birth_date: "2011-11-11",
         created_at: "2019-10-05 08:15:00",
         updated_at: "2019-10-05 08:15:00",
       },
       App\Customer {#3025
         id: 2,
         name: "Alainen Kim",
         birth_date: "2012-12-12",
         created_at: "2019-10-05 13:04:40",
         updated_at: "2019-10-05 13:04:40",
       },
     ],
   }
```

Koko taulun sisältö taulukkona

```
>>> App\Customer::all()->toArray();
=> [
     [
       "id" => 1,
       "name" => "Guru Ken",
       "birth_date" => "2011-11-11",
       "created_at" => "2019-10-05 08:15:00",
       "updated_at" => "2019-10-05 08:15:00",
     ],
     [
       "id" => 2,
       "name" => "Alainen Kim",
       "birth_date" => "2012-12-12",
       "created_at" => "2019-10-05 13:04:40",
       "updated_at" => "2019-10-05 13:04:40",
     ],
   ]
>>>
```

Yksittäinen alkio tai sen kenttä

```
>>> App\Customer::all()[0];
=> App\Customer {#3024
     id: 1,
     name: "Guru Ken",
     birth_date: "2011-11-11",
     created_at: "2019-10-05 08:15:00",
     updated_at: "2019-10-05 08:15:00",
   }
>>> App\Customer::all()[0]->name;
=> "Guru Ken"
```

Ensimmäinen tai viimeinen tietue

```
>>> App\Customer::first();
>>> App\Customer::latest()->first();
```

Kaikkien tietuiden name-kenttä

```
>>> App\Customer::all()->map->name;
=> Illuminate\Support\Collection {#3017
```

```
        all: [
          "Guru Ken",
          "Alainen Kim",
        ],
      }
```

Tietueen etsiminen/hakeminen id-kentän perusteella

```
>>> $customer = App\Customer::find(1);
```

27.08 Tilanne lopuksi: 3 tietuetta

Seuraavaa lukua varten muokkaa tietokantasi sisältö käsittämään lopuksi 3 tietuetta `create()` - ja `delete()` -metodeita käyttämällä. Nimet, id:t ja päivämäärät ovat vapaavalintaisia. Tietokannan sisältö voisi olla esim.

```
mysql> select * FROM customers;
+----+--------------+------------+---------------------+-
| id | name         | birth_date | created_at          | u
+----+--------------+------------+---------------------+-
|  1 | Guru Ken     | 2011-11-11 | 2019-10-05 08:15:00 | 2
|  2 | Vainio Vilja | 2012-12-12 | 2019-10-05 13:04:40 | 2
|  4 | Alainen Kim  | 2010-10-10 | 2019-10-06 05:08:28 | 2
+----+--------------+------------+---------------------+-
3 rows in set (0.00 sec)
```

27.09 Eloquent - tiivistelmä keskeisimmistä

- Tallentaminen 1: Luodaan olio, asetetaan data kenttiin, tallennetaan tietokantaan `save()` -metodilla
- Tallentaminen 1: Asetataan data kenttiin ja samalla tallennetaan tietokantaan `create()` -metodilla
- Päivittäminen 1: Ladataan tietue olioksi, päivitetään haluttu data kenttiin, tallennetaan tietokantaan `save()` -metodilla
- Päivittäminen 2: Käytetään `update()` -metodia
- Poistaminen: Käytetään `delete()` -metodia
- Selaaminen: Lukuisia erilaisia metodeita mm. `all()` -metodi

> Eloquent ORMin avulla tietokannan sisältöä voidaan esittää ja muokata monilla tavoilla. Ks. lisää kokoelmista esim. https://laravel.com/docs/6.x/collections

```
© #AriRantala
```