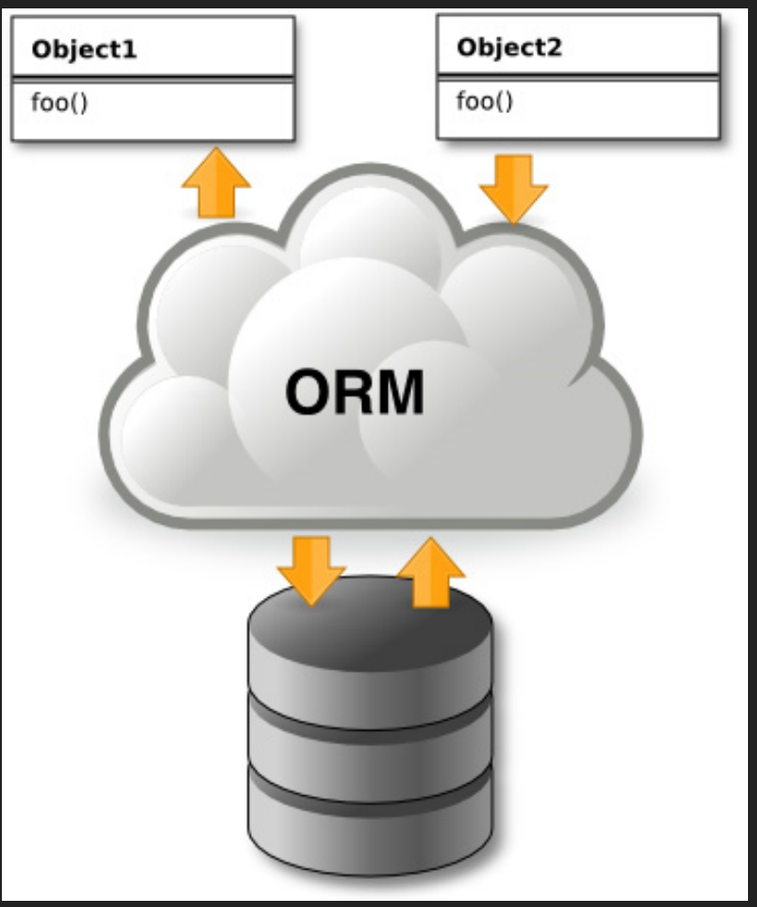
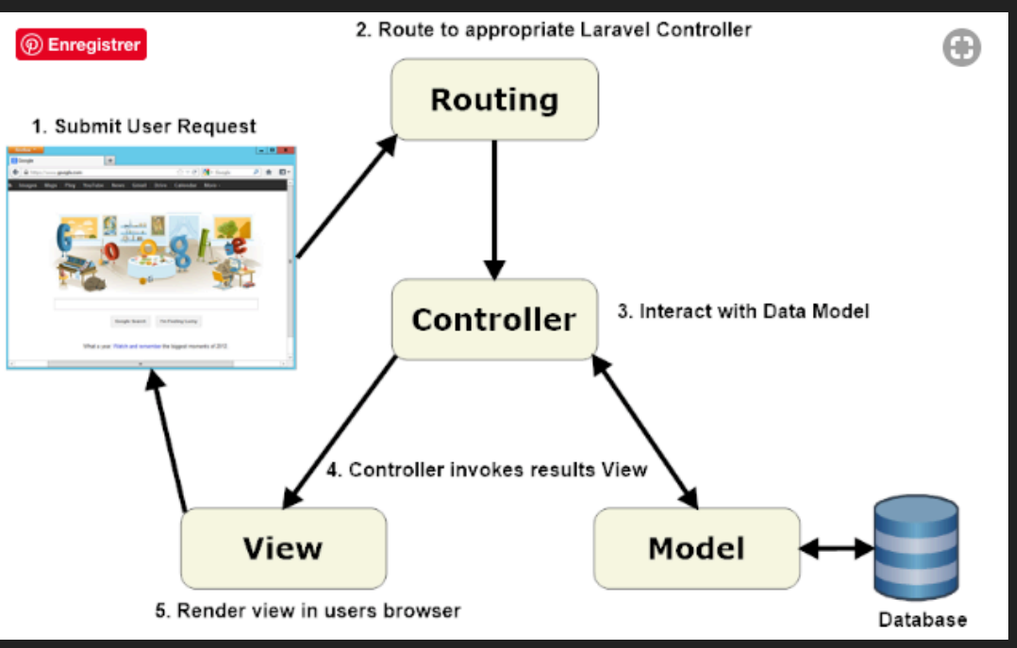
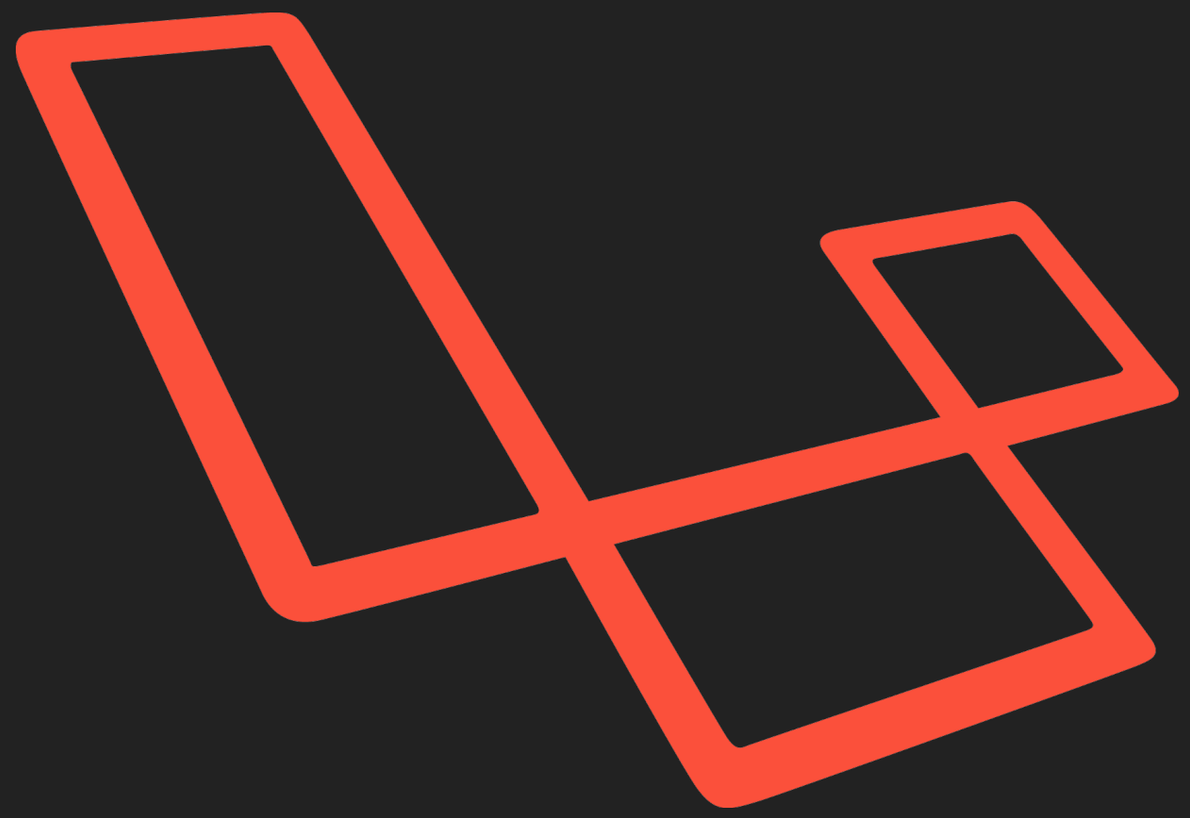
BTS SIO :: SLAM :: PARTIE 3

LARAVEL

v 1.1







CHAP 4

ELOQUENT ORM

UTILISER UN ORM (OBJECT RELATIONAL MAPPING)

- Avec MVC la partie données est traité par le Modèle
- Un modèle est représenté par une classe avec des propriétés et des méthodes
- En général cette classe est liée à une table en base de données.
- Un ORM permet de créer une correspondance entre les classes PHP et la base de données.

INSTALLATION ET INTEGRATION DE MYSQL DANS LARAVEL

- Avant tout de chose il faut installer MySQL sur votre UBUNTU
- Normalement c'est déjà fait lors de l'installation
- Se connecter après à MySql : sudo mysql
- Création de la base : nouvellestar
- Il faudra ensuite changer la configuration du fichier .env qui se trouve à la racine
  Replacer par le bon mot de passe

INSTALLATION ET INTEGRATION DE MYSQL DANS LARAVEL

Step 5 – Create MySQL User and Database

Next, create a MySQL database for your Laravel application. Also create a MySQL user to connect the database from Laravel application. Login to your MySQL server and create MySQL database and user by running following commands.

```
CREATE DATABASE laravel;
CREATE USER 'laravel'@'localhost' IDENTIFIED BY 'secret';
GRANT ALL ON laravel.* to 'laravel'@'localhost';
FLUSH PRIVILEGES;
quit
```

Now edit the **.env** file and update database settings.

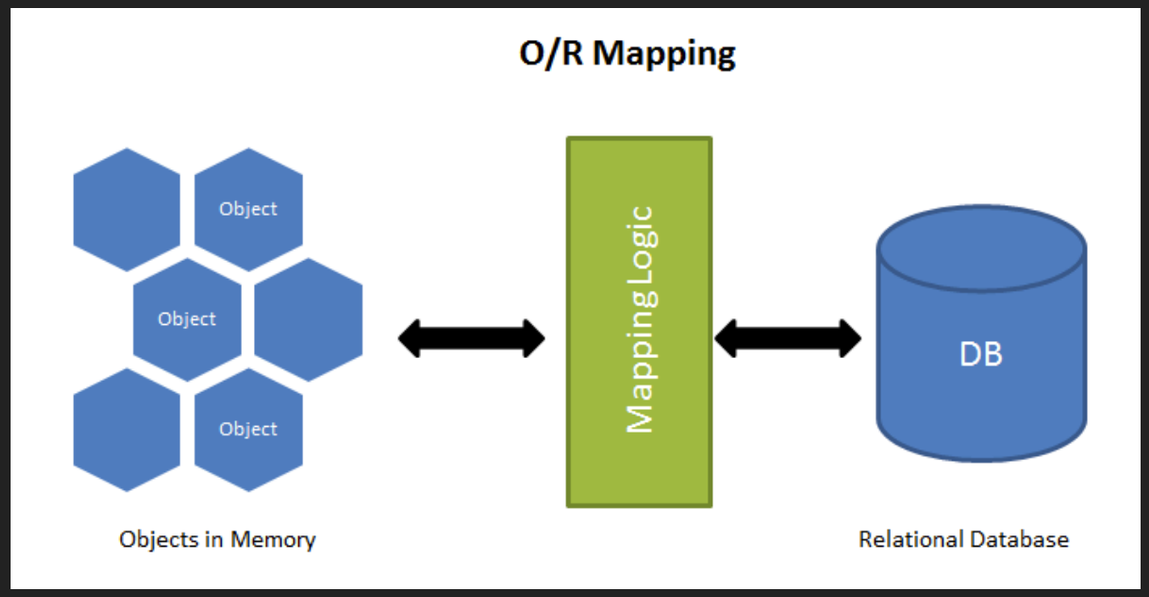
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=laravel
DB_PASSWORD=secret
```

MODÈLES

- Un modèle représente les données et la logique liée à ces données.
- Exemple pour un site e-commerce on pourra avoir :
  - produit
  - commande
  - utilisateur

MODÈLES

- Un modèle est représenté par une classe avec ses propriétés et ses méthodes.
- Ce modèle est généralement lié à une table en BDD
- C'est là qu'intervient l'ORM (Object-Relational-Mapping) celui-ci permet la correspondance entre les classes PHP et la BDD
- Cet ORM s'appelle Eloquent sur LARAVEL



MODÈLES NOMMAGE

- Les modèles comment par une majuscule (c'est une classe)
- Et utilise le CamelCase ex : ArticleJardin
- Toujours au singulier : Article et non ~~Articles~~
- Quand on crée un modèle il est ajouté dans /app
- Au début la classe est sans méthodes, ni propriétés (a vous de compléter) et elle hérite de **Model**

CONSTRUIRE DES MODELES

- Nous allons créer un modèle.
- Pour cela on va utiliser Artisan.
  Artisan est un outil mis à disposition par Laravel.
- Si on veut créer une classe Article on va créer un modèle Article (et non ~~Articles~~)
- php artisan make:model Article (placer vous à la racine de votre site)

```
sebastien@sebastien-VirtualBox:/var/www/laravel$ sudo php artisan make:mode Article
[sudo] Mot de passe de sebastien :
Model created successfully.
sebastien@sebastien-VirtualBox:/var/www/laravel$
```

CONSTRUIRE DES MODELES

- Après cette instruction -> php artisan make:model Article vous devriez avoir dans le répertoire *app/Models*

```
sebastien@sebastien-VirtualBox:/var/www/laravel/app/Models$ ls
Article.php  User.php
sebastien@sebastien-VirtualBox:/var/www/laravel/app/Models$ pwd
/var/www/laravel/app/Models
sebastien@sebastien-VirtualBox:/var/www/laravel/app/Models$
```

- Ce fichier est une classe pour l'instant sans propriétés ni méthodes qui hérite de la classe Model.

```
sebastien@sebastien-VirtualBox:/var/www/laravel/app/Models$ cat Article.php
<?php

namespace App\Models;

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

class Article extends Model
{
    use HasFactory;
}
sebastien@sebastien-VirtualBox:/var/www/laravel/app/Models$
```

CONSTRUIRE DES MODELES

- Cette classe doit être maintenant associé à une table en BDD si l'on veut une persistance des données.
- Cette action s'appelle la **migration**.

```
sebastien@sebastien-VirtualBox:/var/www/laravel$ sudo php artisan make:migration create_table_articles
Created Migration: 2020_12_08_090721_create_table_articles
sebastien@sebastien-VirtualBox:/var/www/laravel$
```

- La commande a créé le fichier de migration (database/migrations). Il faut le personnaliser afin d'indiquer les champs et leur type.
- On aurait pu créer le modèle et faire la migration en une seule instruction : php artisan make.model Article -- migration

FICHIERS DE MIGRATION

```
(base) macbook-de-seb:nouvelle-star sebastien$ cd database/migrations/
(base) macbook-de-seb:migrations sebastien$ 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
2019_11_22_104437_create_table_articles.php
(base) macbook-de-seb:migrations sebastien$
```

```php
<?php

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

class CreateTableArticles extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('table_articles', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->timestamps();
        });
    }

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

MODIFICATION DU FICHIER DE MIGRATION

Par convention :

- le nom des tables sont toujours au pluriel.
- la notation est snake_case / exemple : blog_posts

```php
class CreateTableArticles extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('articles', function (Blueprint $table)
            {
                $table->id();
                $table->string('title');
                $table->text('content');
                $table->timestamps();
        });
    }
```

Pour terminer cette migration taper à la racine du site **sudo php artisan migrate**

```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.04 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.05 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.02 seconds)
Migrating: 2019_11_22_104437_create_table_articles
Migrated:  2019_11_22_104437_create_table_articles (0.02 seconds)
(base) macbook-de-seb:nouvelle-star sebastien$
```

```
Database changed
mysql> show TABLES;
+-------------------------+
| Tables_in_nouvellestar |
+-------------------------+
| articles                |
| migrations              |
| password_resets         |
| users                   |
+-------------------------+
4 rows in set (0.00 sec)
```

UP()

- La classe CreateTableArticles hérite de Migration
- Cette classe sert à créer la table en BDD.
- C'est la méthode up() qui est appelé pour créer la table
- La méthode down() sera étudié plus loin mais pour l'instant elle sert à détruire la table
- L'id est créé par Laravel automatiquement et sert d'identifiant
- C'est un Id séquentiel qui s'auto-incrémente.
- Laravel s'en sert pour récupérer un objet exemple **$my_article = Article::find(12);**

```php
class CreateTableArticles extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('articles', function (Blueprint $table)
            {
                $table->id();
                $table->string('title');
                $table->text('content');
                $table->timestamps();
        });
    }

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

TYPE DE COLONNE – LES ENTIERS

- $table->integer('column_name');
  - Entier signé (positif ou négatif) (sur 4 octets)
- $table->integer('column_name')->unsigned();
  - Entier non signé (positif)
- $table->integer('column_name')->nullable();
  - Dont la valeur peut être NULL
- $table->integer('column_name')->default(4);
  - valeur 4 par défaut
- $table->integer('column_name')
  ->unsigned()
  ->nullable();
  - Entier non signé pouvant être NULL
- $table->tinyInteger('column_name'); // 1 octet
- $table->smallInteger('column_name'); // 2 octets
- $table->mediumInteger('column_name'); // 3 octets
- $table->bigInteger('column_name'); // 8 octets

TYPE DE COLONNE

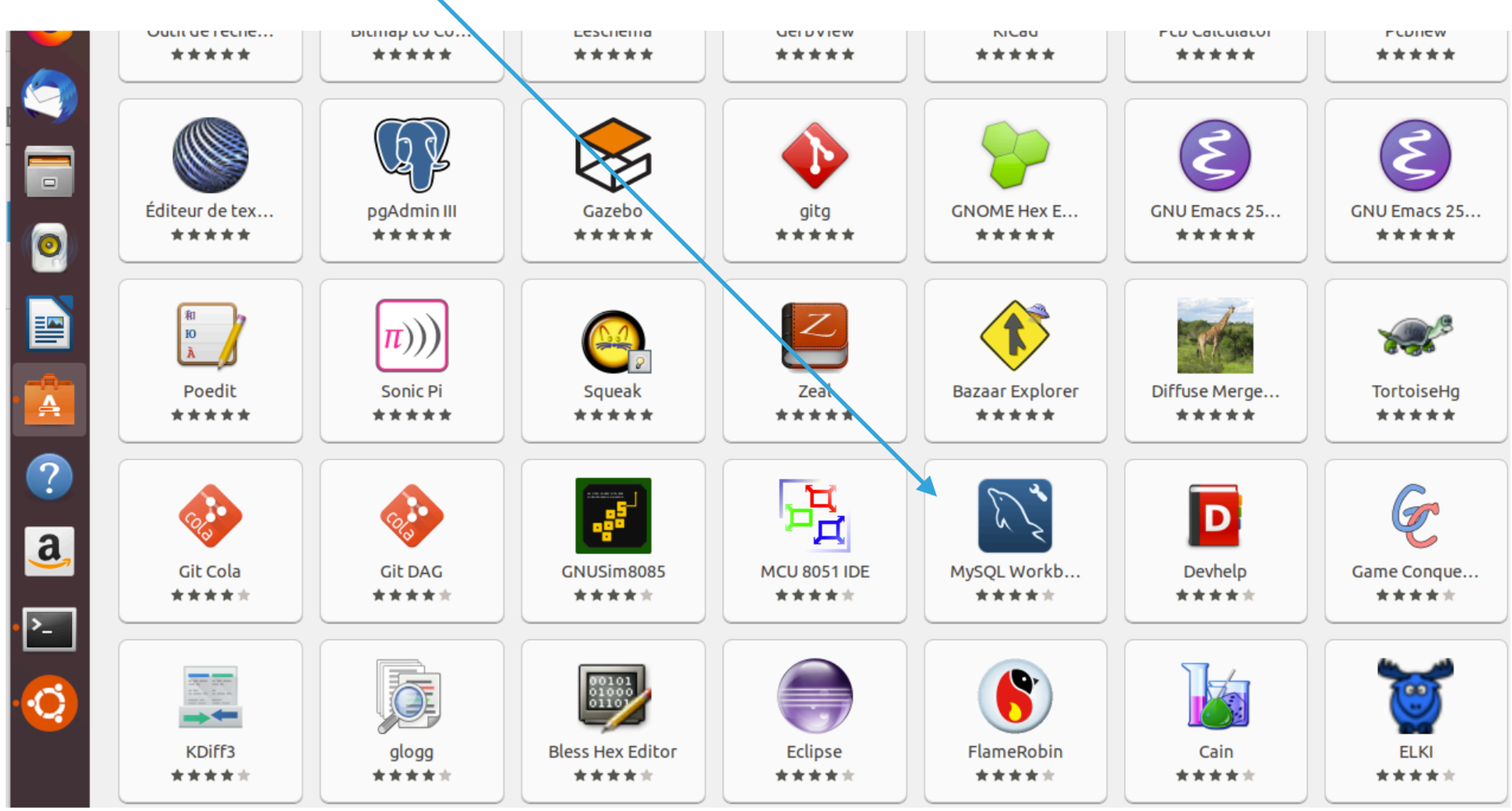
- $table->string('column_name');
  - Chaîne de caractères
- $table->string('column_name', 100);
  - Chaîne de caractères avec une longueur maximale de 100
- $table->boolean('column_name');
  - type booléen
- $table->decimal('column_name', 3, 2);
  - 3 chiffres dont 2 après la virgule
- $table->datetime('column_name');
- $table->date('column_name');
- $table->timestamp('column_name');
- $table->enum('column_name');
- $table->json('column_name');
- $table->increment('column_name');

LARAVEL

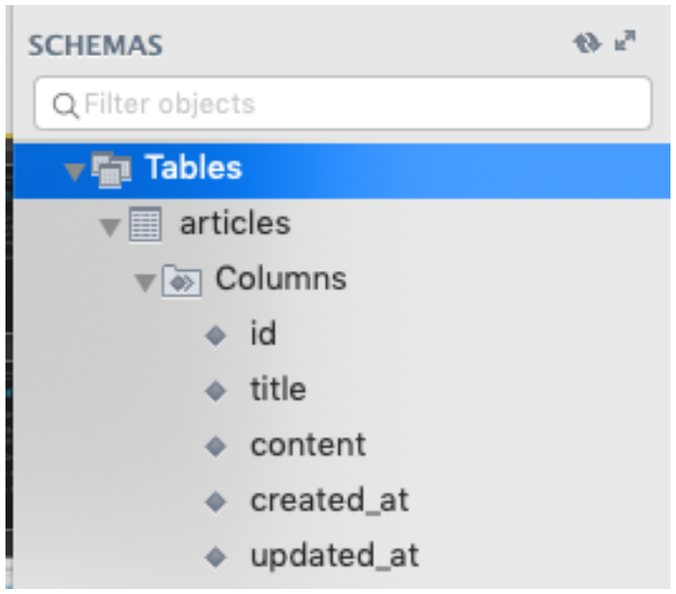
UN PEU DE CONFORT

INSTALLER POUR PLUS DE CONFORT

- Un éditeur de texte comme Sublime Text
- MySQL Work Bench



ON VÉRIFIE QUE NOTRE TABLE



Laravel ajoute les deux derniers champs car on a mis timestamps() Ceci permet de garder une historisation.

TEST AJOUT ARTICLE

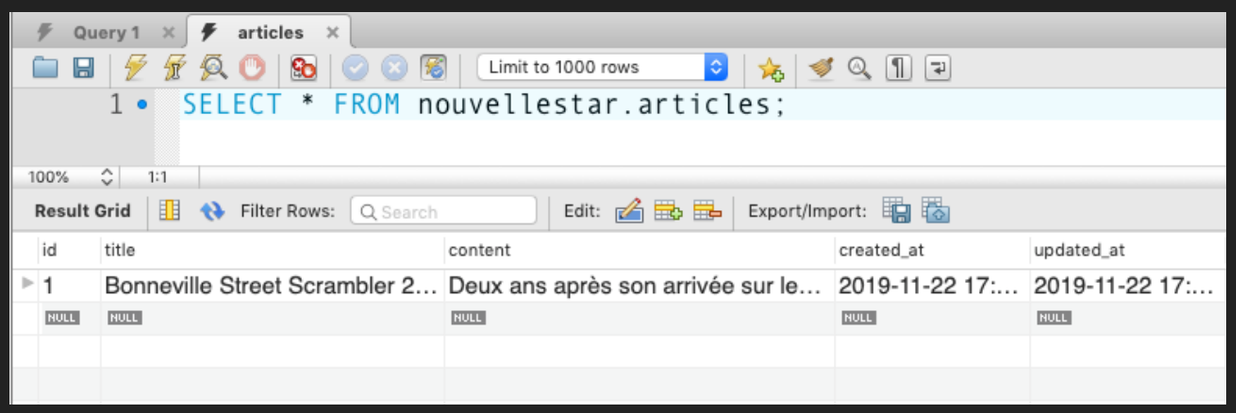
- On va juste ajouter un chemin qui va permettre d'ajouter un article. On fera donc pas de vue pour gagner du temps.

```php
Route::get('/ajoutArticle', function () {
    $article = new Article();
    $article->title="Bonneville Street Scrambler 2019";
    $article->content="Deux ans après son arrivée sur le marché, Triumph remet à jour sa Street
        Scrambler : grâce à un travail sur le moteur, la puissance est en hausse de 10 chevaux
        en passant de 55 à 65 ch, tandis que le régime moteur a été augmenté de 500 tr/mn alors
        que le couple moteur, pour sa part, ne bouge pas. Pour faire face à cet afflux de
        puissance, le freinage, jugé à juste titre un peu timide par le passé, est désormais
        confié à un étrier Brembo à l'avant, tandis que les ressorts de fourche ont été revus
        pour optimiser le comportement routier. Des cartographies moteur (route / pluie) font
        également leur apparition et le contrôle de traction reste déconnectable. Enfin, le
        tableau de bord reçoit une nouvelle finition et la selle un nouveau revêtement. Pour
        ceux qui aiment la personnalisation, Triumph a prévu de nombreux accessoires. Enfin,
        cette machine à la prise en mains facile est bien évidemment disponible en A2.";
    $article->save();
});
```

- Testez…

```php
<?php
use App\Article;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/
```

Pour utiliser la classe Article il vous faudra mettre cette instruction



TEST AJOUT D'ARTICLES

- On va en ajouter plusieurs. Pour cette fois on aura recourt à un tableau avant d'utiliser un formulaire.

```php
Route::get('/ajoutArticles', function () {
    $listeMotos = array('Bonneville Street Twin 2019'=>"Trois ans après son arrivée sur le
        marché, Triumph remet à jour sa Street Twin : grâce à un travail sur le moteur, la
        puissance est en hausse de 10 chevaux en passant de 55 à 65 ch, tandis que le régime
        moteur a été augmenté de 500 tr/mn alors que le couple moteur, pour sa part, ne bouge
        pas.", 'Bonneville T100'=>"La T100, c'est tout un état d'esprit, à commencer par son
        look rétro allié à une conception particulière certes, mais moderne. Le bicylindre
        parallèle et ses carters moteur chromés affiche 865 cm3 et son injection déguisée en
        carburateur, trompe son monde (de connaisseurs), tandis que monte au travers de son
        échappement « Peashooter » un son profond.", 'Bonneville T120' => "La T120, c'est LA
        Bonneville, première du nom en 1959. Bien sûr la nouvelle s'en inspire, et le résultat
        est pour le coup très convaincant. Elle embarque le nouveau bicylindre 1200 à
        refroidissement liquide, à l'instar de la nouvelle Thruxton, alors que celui de
        nouvelle Street Twin, inédit aussi, affiche 900 cm3. Esthétique et finition ne
        souffrent aucun reproche. Le réservoir reprend plus précisément la forme du modèle de
        1968, on retrouve un moyeu de roue arrière conique à l'ancienne, des jantes à rayons
        bien sûr, et le nouveau moteur est juste magnifique (carters en alu brossé et détails
        bronze). ");
    foreach ($listeMotos as $key => $value) {
        $article = new Article();
        $article->title=$key;
        $article->content=$value;
        $article->save();
    }
});
```

TAF

- Vous allez reproduire toutes les étapes vues précédemment afin de créer un modèle **Star**
  - **Création du modèle (Prénom et petite texte de présentation)**
  - **Migration**
  - **Modification du fichier migration pour créer les champs**
  - **etc**
- Vous créer après un chemin afin d'ajouter les stars via un tableau comme nous venons de la faire avec Articles.

AJOUT AVEC UN FORMULAIRE

- On va créer une vue très basique *articles.blade.php*
- Elle affichera à la fois tous les articles et permettra d'en ajouter via un formulaire
- On va ajouter deux routes !
  - Une en **Get** pour l'affichage
  - Une en **Post** pour gérer le formulaire

LA VUE

```blade
<html>
    <head>
        <title>Les Articles</title>
    </head>
    <body>
        <h1>Les articles</h1>

        @if (!count($articles))
            <p>Aucun article en BDD</p>
        @else
            <table>
                <thead>
                    <tr>
                        <th>Identifiant</th>
                        <th>Id de l'utilisateur</th>
                        <th>Nom de l'article</th>
                        <th>Descriptif</th>
                    </tr>
                </thead>
                <tboby>
                @foreach($articles as $article)
                <tr>
                    <td>{{ $article->id }}</td>
                    <td>{{ $article->user_id}}</td>
                    <td>{{ $article->title}}</td>
                    <td>{{ $article->content}}</td>
                </tr>
                @endforeach
                </tboby>
            </table>
        @endif
```

```blade
        <h2>Ajout d'un nouvel article</h2>

        <form method="post">
            {{ csrf_field() }}

            <label for="user_id">User id : </label>
            <input name="user_id" id="user_id"/>
            <br>

            <label for="NomArticle">Nom de l'article : </label>
            <input name="NomArticle" id="title"/>
            <br>

            <label for="Description">Description : </label>
            <textarea name="Description" id="content"></textarea>
            <br>


            <input type="submit" value="Ajouter cet article">
        </form>

    </body>

</html>
```

LES DEUX CHEMINS

```php
Route::get('/articles', function () {
    // Liste des articles du plus récent au plus ancien
    $articles = Article::latest()->get();

    // Affichage de la vue
    return view('articles', ['articles' => $articles]);

});

Route::post('/articles', function () {
    // Enregistrement d'un article
    $article = new Article();
    $article->user_id = request('user_id');
    $article->title= request('NomArticle',"Inconnu");
    $article->content= request('Description',"Non rempli");
    $article->save();

    // Redirection vers la page d'accueil
    return redirect('/articles');

});
```

RESULTATS

Les articles

Identifiant	Id de l'utilisateur	Nom de l'article	Descriptif
4	13	daz	dezed
3	12	Inconnu	Non rempli
2	2	Inconnu	Non rempli
1	2	Article1	Test d'ajout d'article

Ajout d'un nouvel article

User id :

Nom de l'article :

Description :

Ajouter cet article

TAF

- Avec les nouvelles connaissances vous devez modifier la page d'accueil créée à la fin de l'étape 2 (c'est à dire la liste des stars) mais cette fois-ci avec un accès à la base de données.
  - Faire une vue stars.blade.php
- Créer une vue qui permet d'ajouter une star grâce à un formulaire.
- Créer une route de type /stars/{id} qui permettra de récupérer et d'afficher les informations de la star dont l'identifiant sera passé en paramètre via l'URL.
  Vous pourrez utiliser la méthode find()
- Exemple: $ma_star = Star::find(3);

MAJ D'UNE TABLE

- Toutes opérations sur la BDD passe par une migration.
- Pour mettre à jour une table on passe donc par une migration.
- Supposons que l'on souhaite ajouter un champ prix à la table article.

```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan make:migration ajout_articles_prix_column --table articles
Created Migration: 2019_11_23_083920_ajout_articles_prix_column
(base) macbook-de-seb:nouvelle-star sebastien$
```

MAJ D'UNE TABLE

- Voici le fichier migration.
  On peut remarquer qu'on a **table** au lieu de create avec Schema

- On va tout simplement ajouter le champ dans la méthode up()

- Et on fera se destruction dans la méthode down()
  Nous verrons pourquoi juste après.

- Ce qui donne ceci ->

```php
<?php

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

class AjoutArticlesPrixColumn extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('articles', function (Blueprint $table) {
            $table->decimal('prix', 6, 2);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('articles', function (Blueprint $table) {
            $table->dropColumn('prix');
        });
    }
}
```

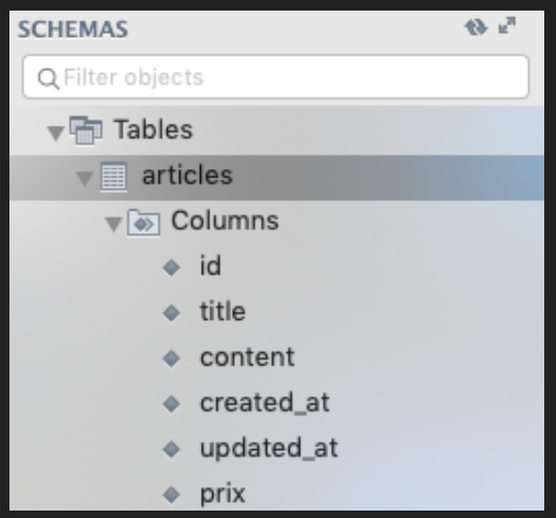
MAJ D'UNE TABLE

- Avant de lancer la migration on peut visualiser les lignes de code que la migration va générer avec : php artisan migrate –pretend

```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan migrate --pretend
AjoutTableArticlesPrixColumn: alter table `table_articles` add `prix` decimal(6, 2) not null
AjoutArticlesPrixColumn: alter table `articles` add `prix` decimal(6, 2) not null
(base) macbook-de-seb:nouvelle-star sebastien$
```

- Si c'est bon on lance la migration.

```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan migrate
Migrating: 2019_11_23_083920_ajout_articles_prix_column
Migrated:  2019_11_23_083920_ajout_articles_prix_column (0.08 seconds)
(base) macbook-de-seb:nouvelle-star sebastien$
```

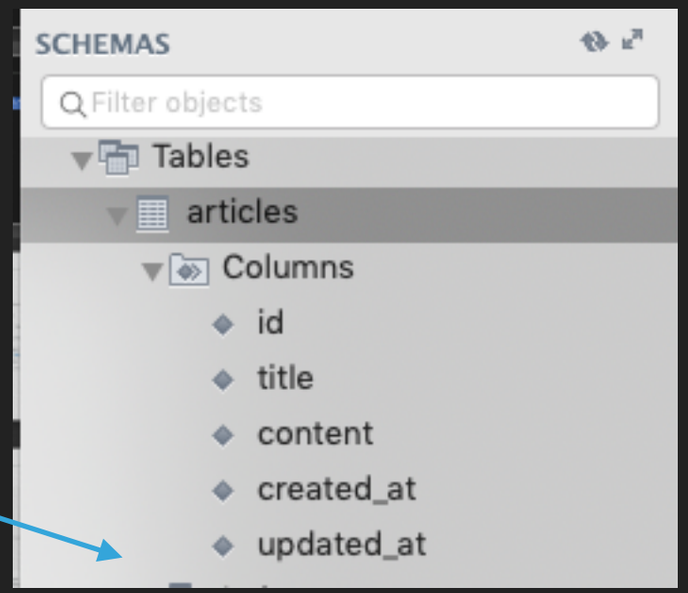


ROLLBACK

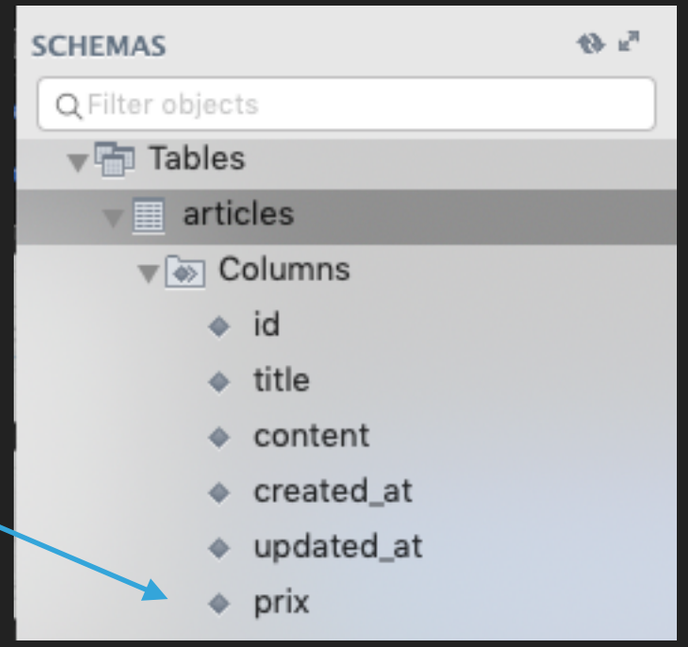
- Il arrive parfois qu'on est fait une erreur lors d'une migration que ce soit à la création ou à la modification.
- C'est là que le **down()** intervient et c'est la raison pour laquelle il fait l'inverse du up() pour revenir à l'état précédent.
- Pour faire ce rollback :
  php artisan migrate:rollback
- Cela annule **la dernière** migration et donc ici l'ajout du champ prix.
- Si vous voulez annuler 3 migrations il faudra le taper 3 fois

ROLLBACK

```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan migrate:rollback
Rolling back: 2019_11_23_083920_ajout_articles_prix_column
Rolled back:  2019_11_23_083920_ajout_articles_prix_column (0.06 seconds)
```




```
(base) macbook-de-seb:nouvelle-star sebastien$ php artisan migrate
Migrating: 2019_11_23_083920_ajout_articles_prix_column
Migrated:  2019_11_23_083920_ajout_articles_prix_column (0.06 seconds)
```



TAF

- En vous inspirant de ce que vous venez de voir. Je vous demande d'ajouter via une migration un champ date_de_naissance à la table stars
- Vérifier que le champ a été bien créé
- Faire un Rollback
- Et revenir à la création de ce champ

ELOQUENT : CONVENTIONS

Clés primaires

- Eloquent suppose que la clé primaire est nommée **id**.
- Il s'en sert avec sa méthode **find**
  - $my_user = User::find(894); (rappel :: pour les méthodes statiques)
- id est de type *entier* et *auto-incrémenté*
- Si vous souhaitez utiliser une clé primaire non incrémentée ou non numérique, vous devez définir la `$incrementing` propriété publique de votre modèle sur `false`. Si votre clé primaire n'est pas un entier, vous devez définir la `$keyType` propriété protected sur votre modèle `string`.

ELOQUENT : CONVENTIONS

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * Indicates if the model should be timestamped.
     *
     * @var bool
     */
    public $timestamps = false;
}
```

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The storage format of the model's date columns.
     *
     * @var string
     */
    protected $dateFormat = 'U';
}
```

- Si vous ne voulez pas des champs created_at et updated_at il faut changer la propriété $timestamps et la mettre à false.
- On peut aussi changer le format des horodatages

ELOQUENT : RÉCUPÉRER LES MODÈLES

```php
<?php

$flights = App\Flight::all();

foreach ($flights as $flight) {
    echo $flight->name;
}
```

```php
$flight = App\Flight::where('number', 'FR 900')->first();
```

```php
$flights = $flights->reject(function ($flight) {
    return $flight->cancelled;
});
```

```php
$flight = App\Flight::find(1);
```

```php
$flights = App\Flight::find([1, 2, 3]);
```

ELOQUENT : RÉCUPÉRER LES MODÈLES

Exceptions Introuvables

Parfois, vous souhaiterez peut-être lancer une exception si aucun modèle n'est trouvé. Ceci est particulièrement utile dans les routes ou les contrôleurs. Les méthodes `findOrFail` et `firstOrFail` vont récupérer le premier résultat de la requête; Cependant, si aucun résultat n'est trouvé, un sera lancé: `Illuminate\Database\Eloquent\ModelNotFoundException`

```
$model = App\Flight::findOrFail(1);

$model = App\Flight::where('legs', '>', 100)->firstOrFail();
```

Si l'exception n'est pas interceptée, une `404` réponse HTTP est automatiquement renvoyée à l'utilisateur. Il n'est pas nécessaire d'écrire des vérifications explicites pour renvoyer les `404` réponses lors de l'utilisation de ces méthodes:

```
Route::get('/api/flights/{id}', function ($id) {
    return App\Flight::findOrFail($id);
});
```

ELOQUENT : RÉCUPÉRER LES MODÈLES

```php
$count = App\Flight::where('active', 1)->count();

$max = App\Flight::where('active', 1)->max('price');
```

Vous pouvez également utiliser les `count`, `sum`, `max` et d' autres méthodes globales fournies par le générateur de requêtes.
Ces méthodes renvoient la valeur scalaire appropriée au lieu d'une instance de modèle complète

ELOQUENT : MAJ

```php
$flight = App\Flight::find(1);

$flight->name = 'New Flight Name';

$flight->save();
```

ELOQUENT : AVEC ARTICLE

```php
Route::get('/articles/{id}', function ($id) {
    // Liste des articles du plus récent au plus ancien
    $article = Article::find($id);
    return "Le nom de cet article est :". $article->title;
});
```

ELOQUENT : SUPPRESSION

Par la clé primaire ::

```php
App\Flight::destroy(1);

App\Flight::destroy(1, 2, 3);

App\Flight::destroy([1, 2, 3]);

App\Flight::destroy(collect([1, 2, 3]));
```

Par requête ::

```php
$deletedRows = App\Flight::where('active', 0)->delete();
```

ELOQUENT : ACCESSEUR

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the user's first name.
     *
     * @param  string  $value
     * @return string
     */
    public function getFirstNameAttribute($value)
    {
        return ucfirst($value);
    }
}
```

ELOQUENT : MUTATEUR

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Set the user's first name.
     *
     * @param  string  $value
     * @return void
     */
    public function setFirstNameAttribute($value)
    {
        $this->attributes['first_name'] = strtolower($value);
    }
}
```

TAF

- Définir un accesseur à star afin d'avoir le nom de la star avec uniquement la première lettre en majuscule
- Définir un mutateur afin d'enregistrer les noms des stars en minuscule.
- Définir un chemin qui permet d'extraire l'ensemble de la table au format JSON .

LARAVEL

LES RELATIONS