## Chapitre 3 – Modéliser les relations simples

### Base de données et mapping relationnel

Créons des modèles qui sont en relation avec d’autres. Par exemple, *Locality* est en relation *OneToMany* avec *Location*. A l’inverse, *Location* est en relation *OneToMany (inverse)* avec *Locality*.

<https://laravel.com/docs/7.x/eloquent-relationships>

1. Modifier le modèle *Locality* de façon à ajouter la relation *OneToMany*.

En Laravel, les relations sont définies sous forme de méthodes d’accès aux données.

//…

class Locality extends Model

{

protected $fillable = ['postal\_code', 'locality'];

protected $table = 'localities';

public $timestamps = false;

/\*\*

\* Get the locations for the locality.

\*/

public function locations()

{

return $this->**hasMany**('App\Location');

}

}

1. Créer le modèle *Location* (et sa relation avec le modèle *Locality*)

**php artisan make:model Location**

*app\Location.php*

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Location extends Model

{

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'slug',

'designation',

'address',

'locality\_id',

'website',

'phone',

];

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'locations';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

/\*\*

\* Get the locality of the location

\*/

public function locality()

{

return $this->**belongsTo**('App\Locality');

}

}

1. Créer et modifier la migration

php artisan make:migration create\_locations\_table

<https://laravel.com/docs/7.x/migrations>

Modifier le fichier généré : ajouter les champs et leurs contraintes. Définir les contraintes de clé étrangère pour le champ *locality\_id*. Ajouter la contrainte d’unicité pour le champ *slug*.

*database\migrations\YYYY\_MM\_DD\_HHMMSS\_create\_locations\_table.php*

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateLocationsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('locations', function (Blueprint $table) {

$table->id();

**$table->string('slug',60)->unique();**

$table->string('designation',60);

$table->string('address',255);

**$table->foreignId('locality\_id');**

$table->string('website',255)->nullable();

$table->string('phone',30)->nullable();

**$table->foreign('locality\_id')->references('id')->on('localities')**

**->onDelete('restrict')->onUpdate('cascade');**

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('locations');

}

}

1. Exécuter les migrations

php artisan migrate

1. Créer et modifier une classe de *seeders*

php artisan make:seeder LocationsTableSeeder

*database\seeds\LocationsTableSeeder.php*

<?php

use Illuminate\Database\Seeder;

**use Illuminate\Support\Str;**

**use App\Location;**

**use App\Locality;**

class LocationsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

Location::truncate();

//Define data

$locations = [

[

**'slug'=>null,**

'designation'=>'Espace Delvaux / La Vénerie',

'address'=>'3 rue Gratès',

**'locality\_postal\_code'=>'1170',**

'website'=>'https://www.lavenerie.be',

'phone'=>'+32 (0)2/663.85.50',

],

[

'slug'=>null,

'designation'=>'Dexia Art Center',

'address'=>'50 rue de l\'Ecuyer',

'locality\_postal\_code'=>'1000',

'website'=>null,

'phone'=>null,

],

[

'slug'=>null,

'designation'=>'La Samaritaine',

'address'=>'16 rue de la samaritaine',

'locality\_postal\_code'=>'1000',

'website'=>'http://www.lasamaritaine.be/',

'phone'=>null,

],

[

'slug'=>null,

'designation'=>'Espace Magh',

'address'=>'17 rue du Poinçon',

'locality\_postal\_code'=>'1000',

'website'=>'http://www.espacemagh.be',

'phone'=>'+32 (0)2/274.05.10',

],

];

//Insert data in the table

foreach ($locations as $data) {

**$locality = Locality::firstWhere('postal\_code',$data['locality\_postal\_code']);**

DB::table('locations')->insert([

**'slug' => Str::slug($data['designation'],'-'),**

'designation' => $data['designation'],

'address' => $data['address'],

**'locality\_id' => $locality->id, //Référence à la table localities**

'website' => $data['website'],

'phone' => $data['phone'],

]);

}

}

}

Notez que nous associons un lieu *Location* à sa localité *Locality* en recherchant la localité par code postal. De plus, nous utilisons la méthode *Str::slug* pour générer automatiquement le champ *slug* sur base de la désignation.

Attention, nous devons aussi modifier la classe *LocalitiesTableSeeder* puisqu’il y a une contrainte de clé étrangère *ON DELETE restrict*.

1. Modifier la classe de *seeders LocalitiesTableSeeder*

Nous désactivons la vérification des contraintes de clés étrangères avant de vider la table, puis nous la réactivons. Attention à ne faire cela qu’en mode développement, pas en production.

//…

**use App\Locality;**

class LocalitiesTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

**DB::statement('SET FOREIGN\_KEY\_CHECKS=0');**

Locality::truncate();

**DB::statement('SET FOREIGN\_KEY\_CHECKS=1');**

//…

1. Exécuter les seeders
   1. Dans le fichier *DatabaseSeeder.php*, il faut d’abord ajouter la classe *LocationsTableSeeder* à la liste des seeders à exécuter :

*database\seeds\DatabaseSeeder.php*

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Seed the application's database.

\*

\* @return void

\*/

public function run()

{

$this->call(**[**

ArtistsTableSeeder::class,

TypesTableSeeder::class,

LocalitiesTableSeeder::class,

RolesTableSeeder::class,

LocationsTableSeeder::class,

**]**);

}

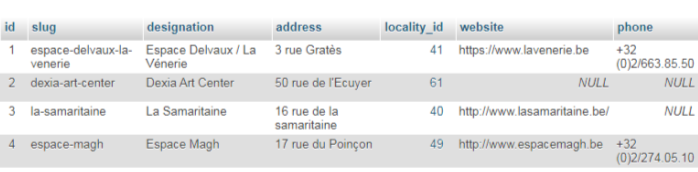
}

* + 1. Ensuite, il faut regénérer le fichier autoloader de Composer !

composer dump-autoload

* + 1. Enfin, nous pouvons exécuter les seeders :

php artisan db:seed



1. Versionner le projet

git status

git add .

git commit -m "Modèle Location +Relation ManyToOne with Locality"

### Contrôleurs et templates

1. Modifier le contrôleur *LocalityController*

Nous voulons à présent afficher une localité et tous les lieux de spectacle qui s’y trouvent. Si ce n’est déjà fait, ajoutons du code dans la méthode *show*.

*App\Http\Controllers\LocalityController.php*

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

**use App\Locality;**

class LocalityController extends Controller

{

//…

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$locality = Locality::find($id);

return view('locality.show',[

'locality' => $locality,

]);

}

//…

}

1. Modifier le template *show.blade.php*

Dans le dossier *locality*, afficher la liste des lieux dans le template *show.blade.php*.

*ressources/views/locality/show.blade.php*

@extends('layouts.app')

@section('title', 'Liste des localités')

@section('content')

<h1>{{ $locality->postal\_code }} {{ $locality->locality }}</h1>

<ul>

@foreach($locality->locations as $location)

<li>{{ $location->designation }}</li>

@endforeach

</ul>

@endsection

1. Enfin, nous devons ajouter une route afin de pouvoir accéder à cette page à travers notre application. La route get('locality/{id}') envoie l’id spécifiée dans l’URL comme paramètre de la méthode *show* du contrôleur *LocalityController*.

*routes/web.php*

<?php

//…

Route::get('artist', 'ArtistController@index');

Route::get('artist/{id}', 'ArtistController@show');

Route::get('type', 'TypeController@index');

Route::get('type/{id}', 'TypeController@show');

Route::get('locality', 'LocalityController@index');

**Route::get(' locality/{id}', 'LocalityController@show');**

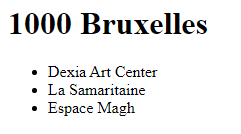
Route::get('role', 'RoleController@index');

Route::get('role/{id}', 'RoleController@show');

* Lancer le serveur interne (ou accéder directement à l’URL de votre Virtual Host)

php artisan serve

* Accéder à l’URL <http://localhost:8000/locality/1> (ou [http://laravel.localhost/locality/](http://laravel.localhost/locality/5)1)



1. Versionner le projet

git status

git add .

git commit -m "Controller Locality +template +route"

### A vous de jouer !

1. **Répétez ces opérations pour les relations entre les modèles *Location, Show* et *Representation*.  
   Notez qu’il y a deux références (clés étrangères) dans la table *representations*.**
   1. Créer le modèle sans oublier sa relation *ManyToOne (OneToMany inverse)*
   2. Créer les migrations
   3. Exécuter les migrations
   4. Générer la classe de seeders
   5. Définir les données test (seeders) sans oublier la référence dans la classe secondaire
   6. Charger les données test (seeders) dans la table
   7. Créer les contrôleurs, les templates et définissez les routes
2. Versionner le projet

Bravo!