

LARAVEL Framework - Tutorial 05 - Anexo

Sumario

- 1 Creación y uso de relaciones Muchos a Muchos M:N (many to many) en Laravel 5
 - ♦ 1.1 Creación de Modelos ejemplo relación many to many en Laravel 5
 - ♦ 1.2 Creación de Migraciones ejemplo relación many to many en Laravel 5
 - ♦ 1.3 Ejemplo de uso de relaciones many to many en Laravel 5

Creación y uso de relaciones Muchos a Muchos M:N (many to many) en Laravel 5

- En algunos caso cuando se esta desarrollando un sistema, se suele encontrar modelos que implementan una relación de muchos a muchos.
- **Estas relaciones en la implementación funcionan con una tabla intermedia que une a las 2 entidades relacionadas.**
- Laravel tiene una sintaxis especial para este tipo de tablas: éstas **deben conservar cierto orden alfabético en su nombre y campos** (Estos pueden personalizarse al momento de establecer la relación en el modelo).

- Supongamos que queremos realizar una aplicación en la que tenemos Aviones y Rutas
 - ♦ Un avión puede realizar 1 o varias rutas.
 - ♦ Una ruta puede ser realizada por 1 o varios aviones.
- **avion_ruta** // Será el nombre de la tabla intermedia por defecto. Este nombre debe estar en orden alfabético, en singular y con un _ entre las entidades que relaciona.

Creación de Modelos ejemplo relación many to many en Laravel 5

- Creación de los **modelos Avion y Ruta**

```
php artisan make:model Avion
# Model created successfully.
# Created Migration: 2015_05_20_205931_create_avions_table
```

```
php artisan make:model Ruta
# Model created successfully.
# Created Migration: 2015_05_20_205940_create_rutas_table
```

- Contenido del fichero **App\Avion.php**:

```
<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Avion extends Model {

    // Nombre de la tabla en MySQL.
    protected $table='aviones';

    // Atributos que se pueden asignar de manera masiva.
    protected $fillable = array('modelo','longitud','capacidad','velocidad','alcance');

    // Aquí ponemos los campos que no queremos que se devuelvan en las consultas.
    protected $hidden = ['created_at','updated_at'];

    // Relación de Aviones y Rutas
    public function rutas()
    {
        // 1 ruta puede ser realizada por 1 o varios aviones
        // Por defecto el nombre de la tabla intermedia debe estar en
        // orden alfabético, en singular y con un _ entre las entidades que relaciona.
        // En este caso la tabla sería: avion_ruta

        // Si se va a utilizar un nombre distinto de tabla se indicaría como segundo parámetro.
        // return $this->belongsToMany('App\Avion','avionesrutas');
```

```

return $this->belongsToMany('App\Ruta')->withTimestamps();
}
}

```

- Contenido del fichero **App\Ruta.php**:

```

<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Ruta extends Model {

    // Nombre de la tabla en MySQL.
    protected $table='rutas';

    // Atributos que se pueden asignar de manera masiva.
    protected $fillable = array('nombre','distancia','nivel_vuelo');

    // Aquí ponemos los campos que no queremos que se devuelvan en las consultas.
    protected $hidden = ['created_at','updated_at'];

    // Relación de Rutas con Avión
    public function aviones()
    {
        // 1 ruta puede ser realizada por 1 o varios aviones
        // Por defecto el nombre de la tabla intermedia debe estar en
        // orden alfabético, en singular y con un _ entre las entidades que relaciona.
        // En este caso la tabla sería: avion_ruta

        // Si se va a utilizar un nombre distinto de tabla se indicaría como segundo parámetro.
        // return $this->belongsToMany('App\Avion','avionesrutas');
        return $this->belongsToMany('App\Avion')->withTimestamps();
    }
}

```

Creación de Migraciones ejemplo relación many to many en Laravel 5

- Contenido del fichero **2015_05_20_205931_create_avions_table.php**:

```

<?php

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

class CreateAvionsTable extends Migration {

    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('aviones', function(Blueprint $table)
        {
            $table->increments('id');
            $table->string('modelo');
            $table->float('longitud');
            $table->integer('capacidad');
            $table->integer('velocidad');
            $table->integer('alcance');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
    }
}

```

```

{
Schema::drop('aviones');
}

}

```

• Contenido del fichero **2015_05_20_205940_create_rutas_table:**

```

<?php

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

class CreateRutasTable extends Migration {

/**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('rutas', function(Blueprint $table)
        {
            $table->increments('id');
            $table->string('nombre');
            $table->integer('distancia');
            $table->integer('nivel_vuelo');
            $table->timestamps();
        });

        // Creamos la tabla pivot de la relación muchos a muchos.
        // La hacemos en esta migración o se podría hacer a parte.

        // Según el orden de creación, la tabla aviones se ejecuta primero y luego
        // vendría esta migración, y así aprovechamos a hacer la tabla que relaciona las otras dos.

        // Utilizamos la convención estándar para nombrar a la tabla pivot:
        // El nombre de la tabla intermedia (pivot) debe estar
        // en orden alfabético, en singular y con un _ entre las entidades que relaciona.

        Schema::create('avion_ruta',function(Blueprint $table)
        {
            $table->integer('avion_id')->unsigned()->index();
            // Programamos que si borramos un avión lo borre también en la tabla pivot
            $table->foreign('avion_id')->references('id')->on('aviones')->onDelete('cascade');

            // Programamos que si borramos una ruta la borre también en la tabla pivot
            $table->integer('ruta_id')->unsigned()->index();
            $table->foreign('ruta_id')->references('id')->on('rutas')->onDelete('cascade');

            $table->timestamps();
        });

    }

/**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('rutas');
        Schema::drop('avion_ruta');
    }

}

```

Ejemplo de uso de relaciones many to many en Laravel 5

- Código de ejemplo del fichero **app/Http/routes.php**:

```
<?php

/*
|-----
| Application Routes
|-----
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

use App\Avion;
use App\Ruta;

Route::get('/', function()
{
    // Creamos dos Aviones
    $avion = new Avion(array('modelo'=>'Iberia Boeing 737', 'longitud'=>'125', 'capacidad'=>'240', 'velocidad'=>'980', 'alcance'=>'12500'));
    $avion->save();
    $avion = new Avion(array('modelo'=>'Ryanair Airbus A320', 'longitud'=>'132', 'capacidad'=>'200', 'velocidad'=>'900', 'alcance'=>'11000'));
    $avion->save();

    // Creamos tres rutas
    $ruta = new Ruta(array('nombre'=>'Santiago-Madrid', 'distancia'=>'430', 'nivel_vuelo'=>'220'));
    $ruta->save();
    $ruta = new Ruta(array('nombre'=>'Santiago-Madrid', 'distancia'=>'480', 'nivel_vuelo'=>'135'));
    $ruta->save();
    $ruta = new Ruta(array('nombre'=>'Santiago-Barcelona', 'distancia'=>'980', 'nivel_vuelo'=>'250'));
    $ruta->save();

    // Al primer avión le asignamos una nueva ruta:
    $ruta = new Ruta(array('nombre'=>'Sevilla-Ibiza', 'distancia'=>'580', 'nivel_vuelo'=>'210'));
    Avion::find(1)->rutas()->save($ruta);

    // Al primer avión le asignamos una ruta ya creada: ruta 1 ('Santiago-Madrid')
    Avion::find(1)->rutas()->attach(1);

    // Al segundo avión le asignamos una ruta ya creada: ruta 3 ('Santiago-Barcelona')
    Avion::find(2)->rutas()->attach(3);

    // Al segundo avión le desasignamos la ruta 3 ('Santiago-Barcelona')
    Avion::find(2)->rutas()->detach(3);

    // Al segundo avión le asignamos las ruta 1, 2, 3 y 4.
    Avion::find(2)->rutas()->sync(array(1,2,3,4));

    // Obtenemos todas las rutas del avión 1.
    $avion = Avion::find(1);

    // dd($avion->rutas->toArray());
    /* Resultado:

        array:2 [?]
          0 => array:5 [?]
            "id" => 4
            "nombre" => "Sevilla-Ibiza"
            "distancia" => 580
            "nivel_vuelo" => 210
            "pivot" => array:2 [?]
              "avion_id" => 1
              "ruta_id" => 4
          ]
        ]
          1 => array:5 [?]
            "id" => 1
            "nombre" => "Santiago-Madrid"
```

```

        "distancia" => 430
        "nivel_vuelo" => 220
        "pivot" => array:2 [?
            "avion_id" => 1
            "ruta_id" => 1
        ]
    ]
}

*/

// Mostramos todas las rutas a 'Santiago-Madrid'
foreach(Ruta::whereNombre('Santiago-Madrid')->get() as $ruta)
{
    echo "Id: $ruta->id - $ruta->nombre <br/>";
}
/*
        Id: 1 - Santiago-Madrid
        Id: 2 - Santiago-Madrid
*/

// Mostramos todos los aviones que hacen la ruta 'Santiago-Madrid'
foreach(Ruta::whereNombre('Santiago-Madrid')->get() as $ruta)
{
    // Esta instrucción y $ruta->aviones()->get() devuelven una colección
    $aviones= $ruta->aviones;

    foreach($aviones as $avion)
    {
        echo "$avion->modelo <br/>";
    }
    /*
            Iberia Boeing 737
            Ryanair Airbus A320
            Ryanair Airbus A320
    */
}

});

```

--Veiga (discusión) 11:39 28 may 2015 (CEST)