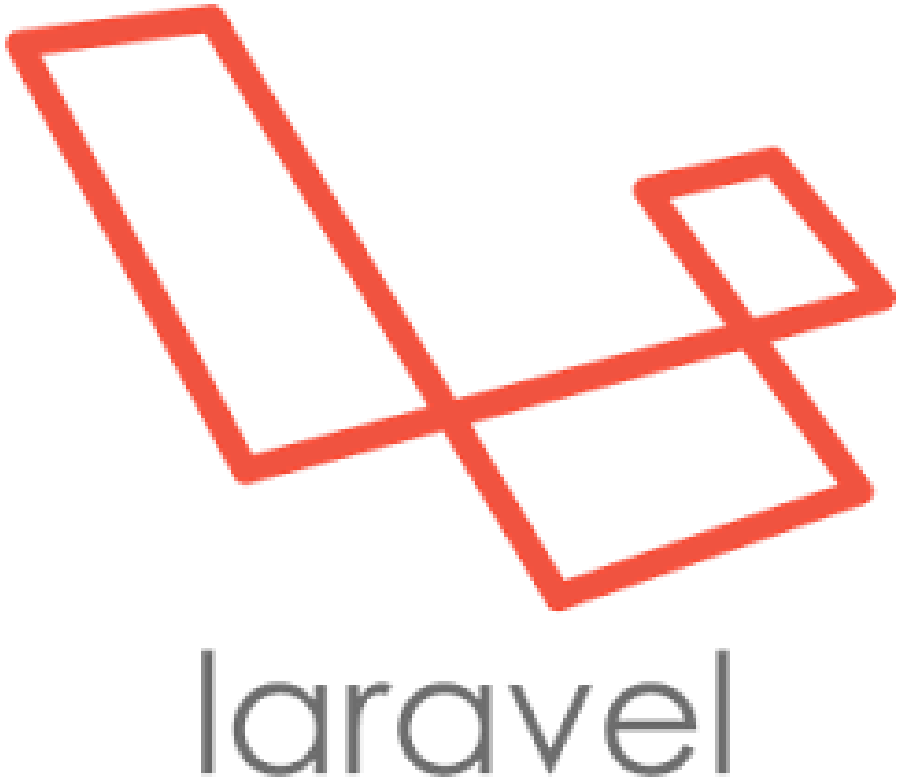
Curso de Laravel



Capítulo 7: El Modelo

Laravel: El Modelo

Eloquent

- Eloquent es el sistema ORM de Laravel
- Posibilita el manejo de datos en la base de datos como si fuesen objetos
- Entrega una forma elegante y simple de interactuar con la base de datos
- Cada Tabla de la base de datos se deberá definir su respectivo modelo. Eloquent se encargará de interactuar entre el código del modelo y la tabla

Laravel: El Modelo

Definición de un modelo

- Por defecto, los modelos se almacenarán en el directorio app
- Para crear un modelo se puede usar el comando make:model de Artisan

```
php artisan make:model Cliente
```

- Este comando creará el archivo Cliente.php al interior de la carpeta app

Laravel: El Modelo

Código generado de un modelo

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    //
}
```

Laravel: El Modelo

Convenciones en Eloquent

- El nombre de los modelos deben estar en singular con la primera letra en mayúscula (ejemplo Producto)
- El nombre de las tablas debe estar en plural (ejemplo: productos)
- Eloquent busca la tabla usando nombre del modelo en minúsculas y en plural en la base de datos. De no encontrarla, arrojará una excepción
- De no usar la convención, deberá en el modelo definirse un atributo con el nombre de la tabla de la forma siguiente:

Laravel: El Modelo

Identificando el nombre de la tabla en el modelo

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    protected $table = "clientes_data";
}
```

Laravel: El Modelo

Clave primaria y timestamps

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    protected $table = "clientes_data";
    //si la PK de la tabla no fuese "id"
    protected $primaryKey = 'id_cliente';
    //si no están los campos "updated_at" y "created_at"
    protected $timestamps = false;
}
```

Laravel: El Modelo

Consultar datos

- Obtener todas las filas de una tabla

```
$clientes =   Cliente::all();

foreach($clientes as $cliente){
   echo $cliente->nombre;
}
```

- Buscar un elemento a partir de su id

```
$clientes = Cliente::find(1);
echo $cliente->name;
```

Laravel: El Modelo

Consultar datos

- Lanzar una excepción cuando no se encuentre un elemento

```
$clientes = Cliente::findOrFail(1);
$clientes = Cliente::where('ventas','>',10000)->firstOrFail();
```

De no encontrarse, se generará una excepción que arrojará un error 404.

Laravel: El Modelo

Consultar datos

- Query Builder con Eloquent

```php
//obtener 10 clientes con ventas de más de 10 mil
$clientes = Cliente::where('ventas','>',10000)->take(10)->get();
//obtener el primer cliente con ventas de más de 10 mil
$clientes = Cliente::where('ventas','>',10000)->first();

//contar los clientes con ventas de más de 10 mil
$clientes = Cliente::where('ventas','>',10000)->count();

//obtener el monto mayor y menor de ventas
$mayorVenta = Cliente::max('ventas');
$menorVenta = Cliente::min('ventas');

//obtener el monto promedio de ventas
$ventasPromedio = Cliente::avg('ventas');
```

Laravel: El Modelo

Insertar un dato

- Para insertar un dato se debe usar el método save()

```
$cliente = new Cliente();
$cliente->nombre = "Juan Perez";
$cliente->save();
```

- Si deseamos obtener el identificador asignado por la BD, se debe acceder al atributo id del objeto insertado

```
$id = $cliente->id;
```

Laravel: El Modelo

Actualizar y borrar registros

- Actualizar un registro

```php
$cliente = Cliente::find(3);
$cliente->email = "cliente@mail.com";
$cliente->save();
```

- Borrar un registro

```php
$cliente = Cliente::find(3);
$cliente->delete();
```

- Borrar más datos

```php
$affectedRows = Cliente::where('ventas','<',10000)->delete();
```

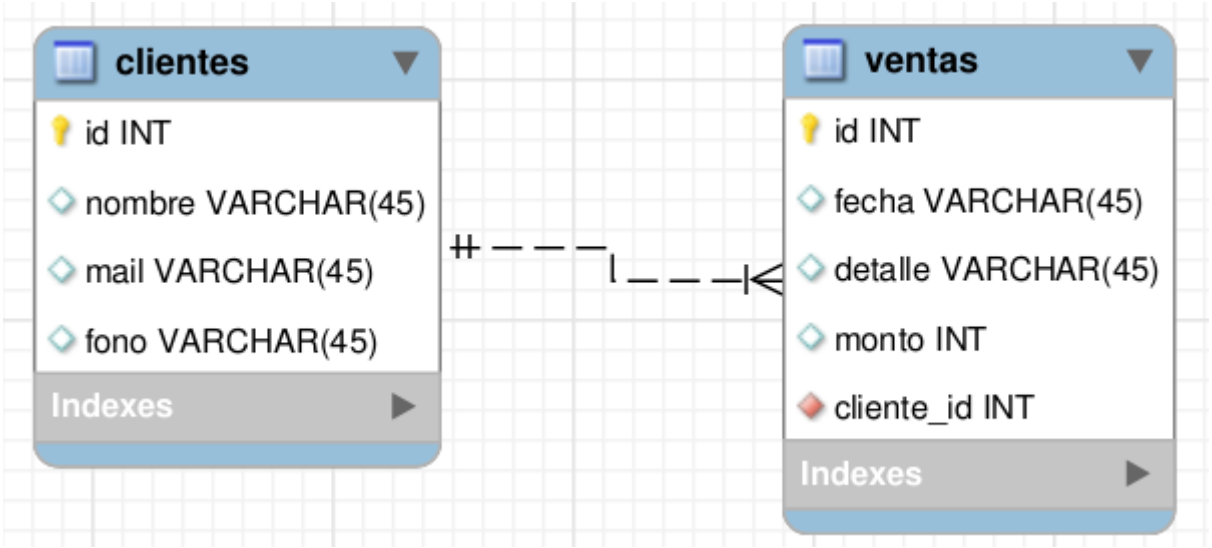
Más documentación en:
https://laravel.com/docs/5.5/eloquent

Laravel: Relaciones

Convenciones de relaciones en Eloquent

- Eloquent utiliza el identificador único id como campo clave para las relaciones

- Para las claves foráneas se debe usar el nombre de la tabla relacionada con el sufijo _id en minúscula



Laravel: Relaciones

Relación uno a uno

- Para una relación uno a uno, se deberá definir una función en el modelo con el método hasOne() indicando la relación como aparece a continuación

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    public function venta()
    {
        return $this->hasOne('App\Venta');
    }
}
```

Laravel: Relaciones

Relación uno a mucho

- Para una relación uno a mucho, se deberá definir una función en el modelo con el método hasMany() indicando la relación como aparece a continuación

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    public function ventas()
    {
        return $this->hasMany('App\Venta');
    }
}
```

Laravel: Relaciones

Relación mucho a uno

- Para una relación mucho a uno, se deberá definir una función en el modelo con el método BelognsTo() indicando la relación como aparece a continuación

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Venta extends Model
{
    public function cliente()
    {
        return $this->BelongsTo('App\Cliente');
    }
}
```

Laravel: Relaciones

Definiendo claves personalizadas

- Para definir claves foráneas diferentes a la convención, se deberán utilizar como parámetro en los métodos de cada relación:

```
return $this->hasOne('App\Venta','clave_foranea','clave_local');
return $this->hasMany('App\Venta','clave_foranea','clave_local');
return $this->BelongsTo('App\Cliente','clave_foranea','clave_local');

//también se puede utilizar solo la clave foranea
//Eloquent considerará el campo id como clave local

return $this->hasOne('App\Venta','clave_foranea');
return $this->hasMany('App\Venta','clave_foranea');
return $this->hasBelongsTo('App\Cliente','clave_foranea');
```

Laravel: Relaciones

Recogiendo registros de un modelo relacionado

- Para recoger registros, se debe hacer llamada a la función definida que retorna la relación como si fuese un atributo del modelo

```
$ventas = Cliente::find(3)->ventas;

foreach($ventas as $venta){
      echo $venta->monto;
}
```

```
$venta = Venta::find(15);
$cliente = $venta->cliente;
echo $cliente->nombre;
```

Laravel: Relaciones

Relación mucho a mucho

- Para una relación mucho a mucho, se deberá definir una función en el modelo con el método belongsToMany() indicando la relación a la tabla intermedia (Ej: ventas)

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    public function Productos()
    {
        return $this->BelongsMany('App\Producto','ventas');
    }
}
```

Laravel: Relaciones

Relación mucho a mucho

- Para obtener los registros de una tabla intermedia, se debe definir el método withPivot() sobre el método BelongsToMany() indicando los campos a retornar

```
return $this->BelongsMany('App\Producto','ventas')
            ->withPivot('fecha_venta','monto');
```

- Para filtrar relaciones condicionando campos en tablas intermediarias se puede usar wherePivot()

```
return $this->BelongsMany('App\Producto','ventas')
            ->wherePivot('estado',1);
```

Laravel: Relaciones

Relaciones

- Para más información sobre las relaciones, Laravel provee documentación oficial en el siguiente enlace:

  https://laravel.com/docs/5.5/eloquent-relationships