MT SOFTECH

SOLUCIONES EN INFORMÁTICA

Eager Loading en Laravel
Problema de las N+1 Consultas

Autor: Ricardo Delgado

¿Qué es?

El problema N+1 es cuando el código ejecuta demasiadas consultas a la base de datos, esto sucede regularmente cuando, a partir de nuestro modelo y el uso del ORM de Eloquent, queremos acceder a cierta relación de nuestro modelo o tabla. Lo que sucede aquí es que iterará N+1 veces para poder conseguir la relación entre una tabla y otra y posteriormente conseguir el dato que queremos mostrar. En resumen, consultará una fila de la base de datos y luego realizará una consulta más para cada registro relacionado, teniendo de esta manera N consultas más el propio registro, N+1.

¿Cómo solucionarlo?

Afortunadamente, Laravel nos provee de una palabra reservada para poder cargar todas estas relaciones en una misma consulta, a esto se le conoce como carga ansiosa o eager loading (ver más en https://laravel.com/docs/8.x/eloquent-relationships#eager-loading). Lo que hace está carga es formar una consulta que contenga estas relaciones, esto en términos de consultas SQL sería utilizar la clásula WHERE seguida del nombre de la llave foránea y después la clausula IN para contener todas las posibles relaciones del mismo modelo.

Ejemplos

Considerando que tenemos un modelo llamado Book que tiene una llave foránea que apunta a un modelo Author, tendríamos lo siguiente:

En el modelo…

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class Book extends Model
{
    public function author()
    {
        return $this->belongsTo(Author::class);
   }
}
```

En el controlador…

```
use App\Models\Book;

$books = Book::all();

foreach ($books as $book) {
    echo $book->author->name;
}
```

Este ciclo ejecutará una consulta para recuperar todos los libros dentro de la tabla de la base de datos, luego otra consulta para cada libro con el fin de recuperar el autor del libro. Entonces, si tenemos 20 libros, el código anterior ejecutaría 21 consultas: una para obtener todos los libros y 20 consultas adicionales para recuperar el autor de cada libro. Esto en términos de consultas SQL quedaría de la siguiente manera:

```
                select * from books
select * from authors where authors.id = 1
select * from authors where authors.id = 2
select * from authors where authors.id = 3
                      .....
select * from authors where authors.id = 20
```

Sin embargo, con el uso de la carga ansiosa (eager loading) quedaría de la siguiente manera:

En el controlador…

```
$books = Book::with('author')->get();

foreach ($books as $book) {
    echo $book->author->name;
}
```

Y como resultado obtendríamos lo siguiente:

```
                select * from books
select * from authors where authors.id in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                    12, 13, 14, 15, 16, 17, 18, 19, 20)
```

Como se puede observar es un cambio drástico, ya que pasamos de tener 21 consultas a solamente 2, con esto le ahorramos recursos al servidor al momento de ejecutar y procesar estas consultas y hacemos que nuestra página web no se ralentice al momento de cargar la página.

Herramientas para depurar las consultas SQL

- Laravel DebugBar:

  Es una herramienta que nos indica las vistas, rutas, consultas y mucho más sobre la página o vista actual en la que estamos. Nos ayuda a tener una perspectiva del proyecto que estamos, muy útil en todo tipo de proyectos, especialmente en proyectos grandes.

- Laravel N+1 Query Detector:

  Como su nombre lo indica, nos ayuda a detectar cuando estamos cayendo en el problema N+1, además de que nos ofrece una solución para poder evitarlo.