Eager Loading

Saat kita mengakses relasi Eloquent sebagai properti, contohnya seperti

```
$article->user->name
```

Maka query untuk artikel tidak akan selesai, atau dia akan menunggu sampai query untuk user selesai, yang mana ini sering disebut dengan `lazy loaded`. Lain dari kata ini juga dikenal dengan ***N+1 query problem***. Namun, karena ada *eager loading* di dalam laravel, ini dapat meredakan hal itu.

Kebetulan pada model artikel ini, kita punya relasi ke model `user`, `category` dan `tags`, dan itu semua kita tampilkan.

Disini saya akan mengilustrasikan nya dengan menunjukkan Anda query pada artikel nya.

```
$articles = Article::query()->limit(9)->get();
```

Kemudian, Anda akan melakukan iterasi terhadap artikel tersebut seperti:

```
@foreach ($article as $item)
    <h4>{{ $article->title }}</h4>
    <p>{{ $article->user->name }} {{ $article->category->name }}</p>

    <ul>
        @foreach ($article->tags as $tag)
            <li>{{ $tag->name }}</li>
        @endforeach
    </ul>
@endforeach
```

Kurang lebih query yang di hasilkan adalah seperti:

```
-- 6:22:41 PM 3.51ms
select `id`, `name` from `categories`

-- 6:22:41 PM 0.67ms
select `id`, `name` from `tags`

-- 6:22:41 PM 1.01ms
select * from `articles` order by `created_at` desc limit 9

-- 6:22:41 PM 0.77ms
select `name`, `slug` from `categories` where exists (select * from
`articles` where `categories`.`id` = `articles`.`category_id`)
```

Maka di sini Anda akan load query sampai **28** kali, **1** untuk artikel, dan **27 / 3** untuk **user**, **kategori**, dan **tag**. Masing-masing dari mereka akan berjumlah **9**, karena kita buat tadi pada aritkel itu `limit(9)`. Artinya, jika Anda limit dia **10**, maka Anda akan load query sampai dengan **31**.

Dan untuk melihat bahwa Anda sedang dalam masalah **N+1 query problem** adalah dengan membuat 1 bari kode tepat di dalam metode `boot` pada `AppServiceProvider` yang ada di `app/Providers/AppServiceProvider.php`.

```
namespace App\Providers;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Pagination\Paginator;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    //...
    public function boot()
    {
        Paginator::useBootstrapFive();
        Model::preventLazyLoading(true);
    }
}
```

Perhatikan bahwa parameter yang diterima oleh `preventLazyLoading` itu adalah `boolean`, yang mana sebenarnya Anda seharusnya membuat dia `true` pada saat `development mode` saja. Jika sudah di `production`, maka itu harusnya Anda buat itu jadi `false`.

Jadi yang menentukan aplikasi kita ini dalam mode `dev` atau `production` adalah dengan membuat value pada file `.env` yang ada di root direktori. Anda bisa perhatikan yang paling atas pada bagian `APP_ENV=local`.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:TCR3oUke6641pCna9kRYl15ep3ksilxCyA7yBizLNRE=
APP_DEBUG=true
APP_URL=http://my-app.test
```

Jadi ketika Anda sudah upload project ini ke server, maka pastikan bahwa `APP_DEBUG` dan `APP_ENV` sudah berubah seperti ini:

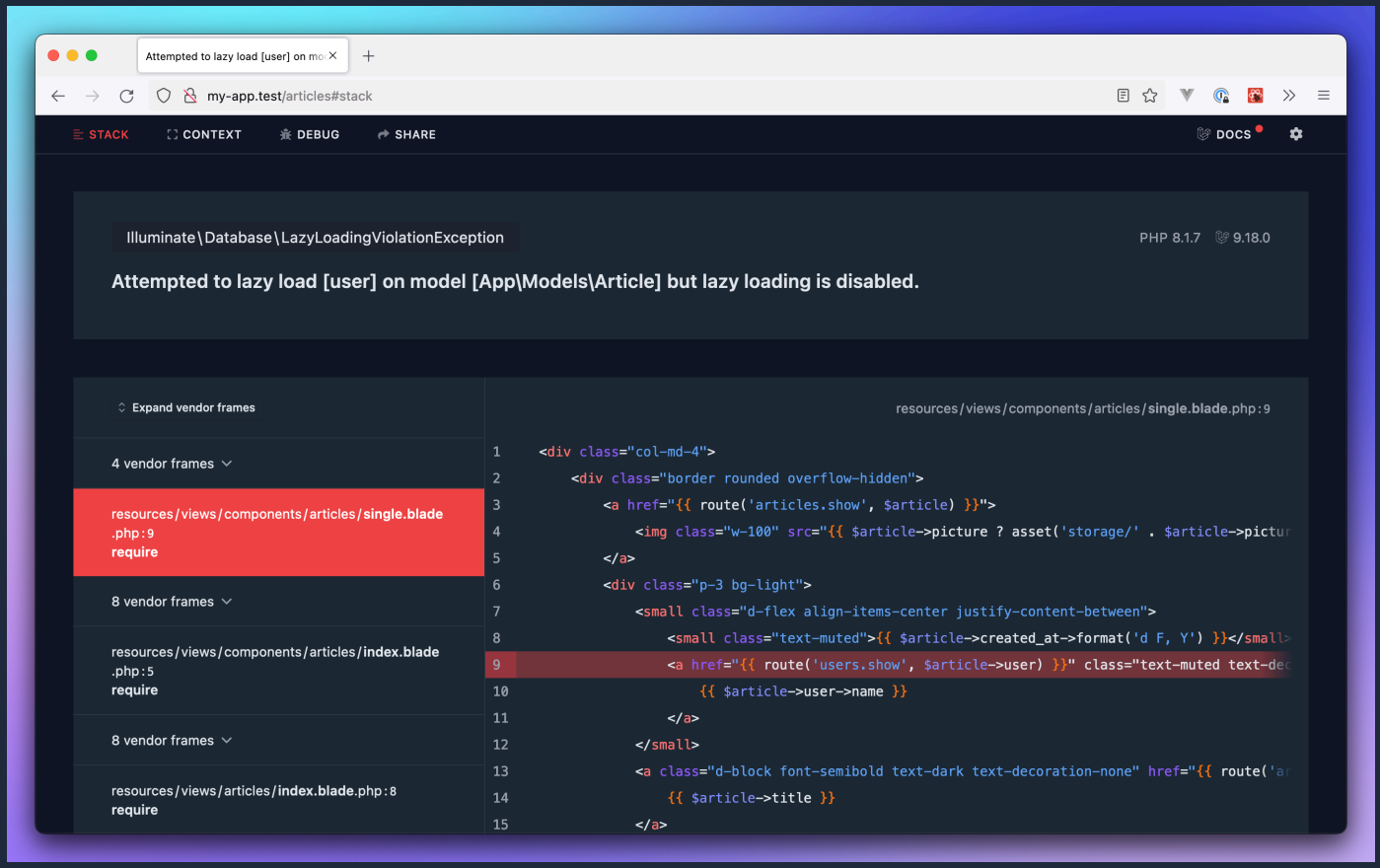
```
APP_NAME=Laravel
APP_ENV=production
APP_KEY=base64:TCR3oUke6641pCna9kRYl15ep3ksilxCyA7yBizLNRE=
APP_DEBUG=false
APP_URL=http://my-app.test
```

Baik, karena kita di dalam development, harusnya yang ada di dalam parameter `preventLazyLoading` sudah benar. Namun, kita tidak ingin merubah-ubah itu sewaktu kita ingin upload project ini ke server. Oleh karena itu, laravel juga sudah mempersiapkan bagaimana cara kita menangani masalah itu dengan memberikan kita helper `app()->isProduction()`.

Maka Anda bisa merubah isi dalam parameter tersebut menjadi seperti:

```
Model::preventLazyLoading(!app()->isProduction());
```

Disini saya tanda `!` yang berarti **jika tidak**. Berarti jelas bahwa dia akan `true` jika sewaktu dalam *development* saja. Dan sekarang jika Anda refresh di browser, Anda akan melihat error yang seperti:



Baik, sekarang Anda bisa buka kembali file `.env` dan ganti `APP_ENV` nya jadi `production` seperti:

```
APP_ENV=production
```

Dan jika Anda refresh kembali di browser, harusnya error nya sudah hilang. Karena kita sedang dalam development, dan kita ingin melihat apa-apa saja query yang berulang. Maka Anda bisa kembalikan itu menjadi local seperti yang tadi

```
APP_ENV=local
```

Baik, sekarang kita akan menghilangkan masalah itu satu per satu. Pertama error nya bilang bahwa `Attempted to lazy load [user] on model [App\Models\Article]`. Untuk itu, mari kita masukkan *eager load* nya tepat pada waktu Anda menampilkan artikel nya pada metode `index` yang ada di `ArticleController.php` yang tadinya kurang lebih masih seperti:

```
public function index()
{
    return view('articles.index', [
        'articles' => Article::latest()->limit(9)->get(),
    ]);
}
```

With

Saya akan jadikan value dari array itu menjadi variabel, sehingga kita bisa bekerja lebih leluasa dan sekalian kita masukkan `with` tepat sebelum `latest` itu seperti:

```
public function index()
{
    $articles = Article::query()
            ->with('user')
            ->latest()
            ->paginate(9);
    return view('articles.index', [
        'articles' => $articles,
    ]);
}
```

Dan sekarang, jika Anda refresh di browser, error nya akan berubah. Yang tadinya itu `Attempted to lazy load [user]` sekarang menjadi `Attempted to lazy load [category]`. Oleh karena itu, mari kita tambah lagi `category` tepat di dalam with itu seperti:

```
Article::query()
    ->with('user', 'category')
    ->latest()
    ->paginate(9);
```

Untuk metode `with` itu bisa merima beberapa paremeter, dan bisa juga menggunakan array seperti:

```
Article::query()
    ->with(['user', 'category'])
    ->latest()
    ->paginate(9);
```

Itu pilihan saja, kalau saya sendiri lebih nyaman jika memakai array. Jadi saya akan tetap menggunakan teknik itu. Dan jika Anda refresh di browser, sekarang error nya akan sudah berganti menjadi `Attempted to lazy load [tags]`. Yang artinya, sekarang Anda harus juga menambahkan `tags` itu tepat pada `with`nya. Jadi sekarang, untuk metode `index` akan sudah seperti ini.

```
public function index()
{
    $articles = Article::query()
            ->with(['user', 'category', 'tags'])
            ->latest()
            ->paginate(9);
    return view('articles.index', [
        'articles' => $articles,
    ]);
}
```

Dan jika Anda refresh di browser, maka harusnya error nya sudah hilang. Tetapi sayang nya itu cukup, karena kita mempunyai beberapa halaman yang itu masih terkena masalah yang namanya `lazy load`. Coba Anda pergi ke halaman kategori dengan mengklik tombol kategori yang ada di artikel-artikel itu, atau bisa langsung kunjungi `http://127.0.0.1:8000/categories/tutorials`. Maka Anda akan mendapatkan masalah yang sama yaitu:

```
Attempted to lazy load [user] on model [App\Models\Article] but lazy loading is disabled.
```

Untuk itu, maka Anda harus pergi lagi ke metode `show` yang ada pada `CategoryController.php` untuk memasukkan `with` tersebut seperti:

```
$category->articles()->with(['user', 'category', 'tags'])-
>paginate(9)
```

Dan masalah akan hilang. Hanya saja, setiap kita ingin menampilkan artikel, maka kita harus ikutkan juga `with` nya. Yang itu harusnya kita lakukan pada beberapa controller seperti `TagController`, `CategoryController`, `ArticleController`, dan `UserController`.

Eager Loading By Default

Sekarang, saya akan mengajari Anda teknik yang namanya *Eager Loading By Default*. Sekarang Anda bisa buka model artikel pada file `app/Models/Article.php`, dan tambahkan *protected* properti untuk `$with` seperti:

```
protected $with = ['user', 'category', 'tags'];
```

Untuk keseluruhan model artikel ini kurang lebih sudah seperti:

```
class Article extends Model
{
    use HasFactory;
    protected $fillable = [...];

    protected $with = ['user', 'category', 'tags'];
```

```
    public function getRouteKeyName() {...}

    public function user() {...}

    public function category() {...}

    public function tags() {...}
}
```

Dan sekarang, Anda bisa hapus semua eager loading yang sudah kita buat pada metode `index` pada `ArticleController` dan `CategoryController` seperti:

```
public function index()
{
    $articles = Article::query()->latest()->limit(9)->get();
    return view('articles.index', [
        'articles' => $articles,
    ]);
}

// CategoryController.php
public function show(Category $category)
{
    return view('categories.show', [
        'category' => $category,
        'articles' => $category->articles()->paginate(9),
    ]);
}
```

Dan sekarang, Anda refresh di browser, mau di halaman mana saja. Error bakal sudah tidak ada lagi, karena kita sudah membuat *eager loading* pada model parent nya yaitu `Article.php`.

Without

Tapi terkadang, ada kalanya kita ingin menampilkan tidak dengan relasinya. Mungkin Anda bisa saja tidak menunjukkan itu tepat pada views nya, tetapi *behind the scene*, query akan tetap membuat tags itu. Nah untuk itu, Anda bisa menggunakan fungsi `without` seperti layaknya Anda menggunakan `with`.

```
Article::query()->without(['tags'])->latest()->get();
```

Karena Anda sudah membuat without, itu artinya Anda tidak akan menampilkan *tags* sewaktu menampilkan artikelnya.

> *"{notice} The book is written by @irsyadadl and power by Parsinta.com"*