Search On N+1 Problem

### **N+1 Problem in Laravel**

في Laravel، الـ N+1 Problem بتحصل لما تطلب بيانات رئيسية (زي كل المستخدمين) وبعدين تبدأ تطلب البيانات المرتبطة بيهم (زي البوستات) جوه لوب، فـ Laravel بينفذ query واحده عشان يجيب البيانات الرئيسية وبعد كده ينفذ query لكل سجل عشان يجيب البيانات المرتبطة بيه. لو عندك مثلًا 50 مستخدم وكل واحد عنده بوستات، ممكن تلاقي نفسك عملت 51 query (واحد للمستخدمين و50 للبوستات) وده بيأثر على أداء التطبيق بشكل سلبي.

### **مثال**

لو عندك الكود ده:

$users = User::all(); foreach ($users as $user) { echo $user->posts; }

الكود ده هيعمل استعلام لكل مستخدم عشان يجيب البوستات بتاعته، فلو عندك 50 مستخدم هتعمل 51 query.

### **حل المشكلة: Eager Loading**

عشان تتجنب المشكلة دي، ممكن تستخدم الـ Eager Loading عن طريق with()، وده بيخلي Laravel يجيب كل المستخدمين ومعاهم البوستات بتاعتهم في 2 queries بس:

$users = User::with('posts')->get(); foreach ($users as $user) { echo $user->posts; }

بالطريقة دي، بدل ما تعمل 51 query، Laravel هيعمل query واحده للمستخدمين وquery واحده للبوستات، وده بيزود سرعة الأداء.

### **Middleware في Laravel: Before and After**

الـ Middleware في Laravel بيستخدم عشان نتحكم فال requests وال responses اللي جيالنا.

#### **Before Middleware**

ده بيشتغل قبل ما الrequest يوصل للـ controller. مثال بسيط هو login:

public function handle($request, Closure $next) {

if (!auth()->check()) {

return redirect('login'); }

return $next($request);

}

لو المستخدم مش مسجل دخول، الـ Middleware هيرجعه لصفحة تسجيل الدخول قبل ما يوصل للـ controller.

#### **After Middleware**

ده بيشتغل بعد ما الـ controller يرجع الـ response. مثلًا لو عاوز تضيف حاجة للـ response زي تعديل الـ headers أو إضافة لوج للوقت:

public function handle($request, Closure $next)

{ $response = $next($request);

$response->headers->set('X-Header-Example', 'Value');

return $response; }

في المثال ده، الـ Middleware بيعدل في الـ headers بعد ما الطلب يتعالج في الـ controller، وده مثال على استخدام After Middleware في تحسين الاستجابة أو إضافة بيانات إضافية للـ response​