Course - Laravel Framework

Rate Limit

Laravel có một phương pháp đơn giản trong việc giới hạn lượng request trong thời gian hẹp, đó là kết hợp với bộ nhớ cache của ứng dụng của bạn để giới hạn bất kỳ hành động nào trong một khoảng thời gian cụ thể.

Tags: rate limit, ti le gioi han, laravel

Giới thiệu

Laravel có một phương pháp đơn giản trong việc giới hạn lượng request trong thời gian hẹp, đó là kết hợp với bộ nhớ cache của ứng dụng của bạn để giới hạn bất kỳ hành động nào trong một khoảng thời gian cụ thể.

> Nếu bạn quan tâm đến việc giới hạn tốc độ các yêu cầu HTTP đến, vui lòng tham khảo tài liệu về phần mềm trung gian rate limiter.

Cấu hình bộ nhớ đệm

Thông thường, rate limiter sử dụng bộ đệm ứng dụng mặc định của bạn theo như khai báo trong `default` key trong tập tin cấu hình `cache` của ứng dụng của bạn. Tuy nhiên, bạn có thể chỉ định driver bộ nhớ đệm mà mà bạn muốn cho rate limiter sử dụng bằng cách khai báo `limiter` key trong tập tin cấu hình cache của ứng dụng của bạn:

```
'default' => 'memcached',

'limiter' => 'redis',
```

Cách sử dụng cơ bản

Facade `Illuminate\Support\Facades\RateLimiter` có thể được sử dụng để tương tác với rate limiter. Phương pháp đơn giản nhất được cung cấp bởi rate limiter là phương thức `attempt`, phương thức này giới hạn một callback nào đó trong một vài giây nhất định.

Phương thức `attempt` sẽ trả về `false` khi callback không còn lần thử nào nữa; nếu không, phương thức `attempt` sẽ trả về kết quả của callback hoặc `true`. Đối số đầu tiên được phương pháp thử chấp nhận là "khóa" giới hạn tỷ lệ, có thể là bất kỳ chuỗi nào bạn chọn đại diện cho hành động bị giới hạn tỷ lệ:

```
use Illuminate\Support\Facades\RateLimiter;

$executed = RateLimiter::attempt(
  'send-message:'.$user->id,
  $perMinute = 5,
```

```
  function() {
    // Send message...
  }
);

if (! $executed) {
  return 'Too many messages sent!';
}
```

Gia tăng theo cách thủ công

Nếu bạn muốn tương tác thủ công với rate limiter, thí có rất nhiều phương pháp khác nhau. Ví dụ: bạn có thể gọi phương thức `tooManyAttempts` để xác định xem rate limiter nào đó có vượt quá số lần thử tối đa được phép mỗi phút hay không:

```
use Illuminate\Support\Facades\RateLimiter;

if (RateLimiter::tooManyAttempts('send-message:'.$user->id, $perMinute = 5)) {
  return 'Too many attempts!';
}
```

Ngoài ra, bạn có thể sử dụng phương thức `remaining` để lấy số lần thử còn lại cho một khóa nhất định. Nếu một khóa nào đó vẫn còn các lần thử lại, bạn có thể gọi phương thức `hit` để tăng tổng số lần thử:

```
use Illuminate\Support\Facades\RateLimiter;

if (RateLimiter::remaining('send-message:'.$user->id, $perMinute = 5)) {
  RateLimiter::hit('send-message:'.$user->id);

  // Send message...
}
```

Xác định sự sẵn có của limiter

Khi một khóa không còn lần thử nào nữa, thì phương thức `availableIn` trả về số giây

còn lại cho đến khi lại có thêm các lần thử khác:

```
use Illuminate\Support\Facades\RateLimiter;

if (RateLimiter::tooManyAttempts('send-message:'.$user->id, $perMinute = 5)) {
  $seconds = RateLimiter::availableIn('send-message:'.$user->id);

  return 'You may try again in '.$seconds.' seconds.';
}
```

Xóa đi số lần thử

Bạn có thể đặt lại số lần thử cho một rate limiter nhất định bằng cách sử dụng phương thức `clear`. Ví dụ: bạn có thể đặt lại số lần thử khi người nhận đọc một tin nhắn nhất định:

```
use App\Models\Message;
use Illuminate\Support\Facades\RateLimiter;

/**
 * Mark the message as read.
 *
 * @param  \App\Models\Message  $message
 * @return \App\Models\Message
 */
public function read(Message $message)
{
  $message->markAsRead();

  RateLimiter::clear('send-message:'.$message->user_id);

  return $message;
}
```