Laravel Cache

Tổng quan về Cache

Trong dự án của các bạn chắc hẳn sẽ có các service, action nào đó hoạt động tốn rất nhiều CPU hoặc rất nhiều thời gian để thực thi xong. Khi gặp phải trường hợp này các bạn có thể xem xét đến sử dụng cache để lưu lại kết quả của hành động vừa rồi để trả về cho lần gọi tiếp theo, nếu dữ liệu không nhất thiết cần realtime. Bộ nhớ lưu trữ cache ở đây có thể là **Memcached** hoặc **Redis**,...

Trong Laravel đã cung cấp cho bạn một API để các bạn có thể làm việc với hầu hết các bộ nhớ Cache store phổ biết hiện nay và đương nhiên là cũng sẽ rất dễ dùng.

Cấu hình

Để cấu hình các thông số liên quan đến cache trong Laravel các bạn có thể chỉnh sửa trong file `config/cache.php`. Mặc định Laravel sẽ sử dụng cache driver là file, khi sử dụng driver này thì các cache data sẽ được lưu trữ vào các file và mặc định chúng sẽ được lưu trữ vào trong thư mục storage/framework/cache/data. Tuy nhiên, nếu như bạn cần một hiệu suất lớn hơn, nhanh hơn thì bạn nên xem xét sử dụng các driver chuyên dụng như Redis, MemCached,...

Database

Khi các bạn sử dụng database làm driver lưu trữ cache thì bắt buộc trong database đó phải có table cache với 3 column lần lượt là `key`, `value`, `expiration`.

Bạn cũng có thể tạo ra migration cho cache table bằng command sau:

```
php artisan cache:table
```

Schema table sẽ có cấu trúc như sau:

```
Schema::create('cache', function ($table) {
    $table->string('key')->unique();
    $table->text('value');
    $table->integer('expiration');
});
```

MemCached

Đối với driver này bạn cần phải cài đặt thêm php extension memcahed. Sau đó bạn cấu hình các thông tin liên quan đến Memached trong file `config/cache.php` key `memcached`

```
'memcached' => [
    'driver' => 'memcached',
    'persistent_id' => env('MEMCACHED_PERSISTENT_ID'),
    'sasl' => [
        env('MEMCACHED_USERNAME'),
        env('MEMCACHED_PASSWORD'),
    ],
    'options' => [
        // Memcached::OPT_CONNECT_TIMEOUT => 2000,
    ],
    'servers' => [
        [
            'host' => env('MEMCACHED_HOST', '127.0.0.1'),
            'port' => env('MEMCACHED_PORT', 11211),
            'weight' => 100,
        ],
    ],
]
```

Redis

Đối với driver này bạn sẽ có 2 sự lựa chọn là sử dụng php extension redis (khuyên dùng) hoặc predis/predis php package. Sau đó bạn có thể cấu hình các thông tin kết nối vào trong file `config/cache.php` key `redis`

Sử dụng

Trong Laravel, bạn có thể sử dụng Cache qua 2 cách là sử dụng Cache facede (`Illuminate\Support\Facades\Cache`) hoặc sử dụng `cache()` helper function (bản chất như nhau).

Để lưu trữ một giá trị nào đó vào cache bạn có thể sử dụng phương thức `put` với cú pháp:

```
use Illuminate\Support\Facades\Cache;

Cache::put($key, $value, $timeToLife);
```

Trong đó:

- `$key` là key của cache đó.

- `$value` là giá trị cần cache vào key đó.
- `$timetoLife` là thời gian sống của cache (tính theo đơn vị giây).

VD: Lưu trữ giá trị vào cache key là "`domain`" sống trong 10 phút.

```
use Illuminate\Support\Facades\Cache;

Cache::put('domain', 'unicode.vn', 600);
```

Để lưu trữ một vào cache vĩnh viễn, bạn có thể sử dụng phương thức `forever`

```
Cache::forever('domain');
```

Để lấy ra một giá trị đang được lưu trữ cache, các bạn sử dụng phương thức `get` với cú pháp:

```
use Illuminate\Support\Facades\Cache;

Cache::get($key, $default);
```

Trong đó:

- `$key` là key của cache các bạn cần lấy ra.
- `$default` là giá trị mặc định sẽ trả về nếu cache không tồn tại hoặc đã hết hạn. Mặc định thì default sẽ là `null`.

VD: Lấy ra giá trị của key "`domain`".

```
$domain = Cache::get('domain');
```

Đối với trường hợp các bạn có muốn sử lí logic để trả về giá trị khi cache không tồn tại, bạn có thể dử dụng tham số thứ 2 là một closure function.

```
$domain = Cache::get('domain', function () {
    return DB::table('options')->where('key', 'domain')->first()->value;
```

```
});
```

Nếu bạn muốn kiểm tra một key nào đó có tồn tại trong cache hay không bạn có thể sử dụng phương thức `has`

```
if (Cache::has('domain')) {
    // tồn tại
}
```

Bạn cũng có thể imcrement/ decrement giá trị của một key nào đó trong cache bằng cách sử dụng phương thức `imcrement` và `decrement`

```
Cache::increment('visited'); // tăng 1 đơn vị.
Cache::increment('visited', 10); // tăng 10 đơn vị.

Cache::decrement('remain'); // giảm 1 đơn vị.
Cache::decrement('remain', 10); // giảm 10 đơn vị.
```

Bạn cũng có thể lấy giá trị cache data và lưu trữ chúng nếu cache data đó không tồn tại, bằng cách sử dụng phương thức `remember` với cú pháp:

```
Cache::remember($key, $timeToLife, function () {
    // ...
});
```

Trong đó:

- `$key` là key của cache đó.
- `$timetoLife` là thời gian sống của cache (tính theo đơn vị giây).
- `closure` function sẽ là nơi chứa logic để tính toán ra giá trị cần lưu trữ vào cache driver khi $key không tồn tại.

```
$domain = Cache::remember('domain', 10 , function () {
    return DB::table('options')->where('key', 'domain')->first()->value;
});
```

Tương tự, bạn có thể sử dụng phương thức `pull` để lấy ra data của cache và đồng thời xóa luôn nó khỏi cache driver.

```
$domain = Cache::pull('domain');
```

Để xóa một cache data các bạn có thể sử dụng phương thức `forget` với giá trị truyền vào là key của cache data.

```
Cache::forget('domain');
```

Hoặc các bạn có thể sử dụng lại phương thức put với tham số `$timeToLife` là 0 hoặc một số âm.

```
Cache::put('domain', 'value', 0);

Cache::put('domain', 'value', -5);
```

Bạn cũng có thể xóa toàn bộ dữ liệu trong cache bằng cách sử dụng phương thức `flush`

```
Cache::flush();
```

Trong một số trường hợp các bạn muốn lưu trữ cache data vào một driver cụ thể các bạn có thể sử dụng phương thức store để định danh driver các bạn muốn lưu trữ.

```
Cache::store('file')->put('domain', 'unicode.vn', 10);
```

Cache Helper

Ngoài việc sử dụng Cache Class, bạn có thể sử dụng Cache Helper do Laravel cung cấp sẵn

```
$value = cache('key');
```

Hàm này sẽ trả về giá trị của cache với key đã khai báo

Nếu truyền vào là 1 mảng và thời gian hết hạn, cache sẽ được tạo với thời gian chỉ định

```
cache(['key' => 'value'], $minutes);
cache(['key' => 'value'], Carbon::now()->addSeconds(10));
```

Nếu hàm `cache()` được gọi không truyền đối số sẽ trả về instance. Lúc này bạn có thể gọi đến các phương thức như Cache Class

```
cache()->remember('users', $seconds, function () {
    return DB::table('users')->get();
});
```

Cache Tags

Cache tag cho phép bạn tag các item liên quan tới nhau trong cache và sau đó flush hết các giá trị cache mà có chung một tag. Bạn có thể truy xuất vào một cache được tag bằng cách truyền vào một mảng các tên tag.

```
Cache::tags(['people', 'artists'])->put('John', $john, $minutes);

Cache::tags(['people', 'authors'])->put('Anne', $anne, $minutes);
```

Để lấy một cache item được tag, truyền vào list tương tự các tag trong phương thức `tags` và sau đó gọi phương thức `get` với key bạn muốn nhận:

```
$john = Cache::tags(['people', 'artists'])->get('John');

$anne = Cache::tags(['people', 'authors'])->get('Anne');
```

Bạn có thể flush tất cả các item có chung một tag hoặc danh sách các tag

Ví dụ, mã lệnh sau sẽ xoá tất cả các cache được tag với giá trị là `people`, `authors`, hoặc cả hai. Vì vậy, cả Anne và John sẽ bị xóa khỏi cache:

```
Cache::tags(['people', 'authors'])->flush();
```

Ngược lại, câu lệnh này sẽ chỉ xoá các cache có tag là `authors`, vì vậy `Anne` sẽ bị xoá, còn `John` thì không:

```
Cache::tags('authors')->flush();
```