RESTful API 제대로 개발하기

by Appkr

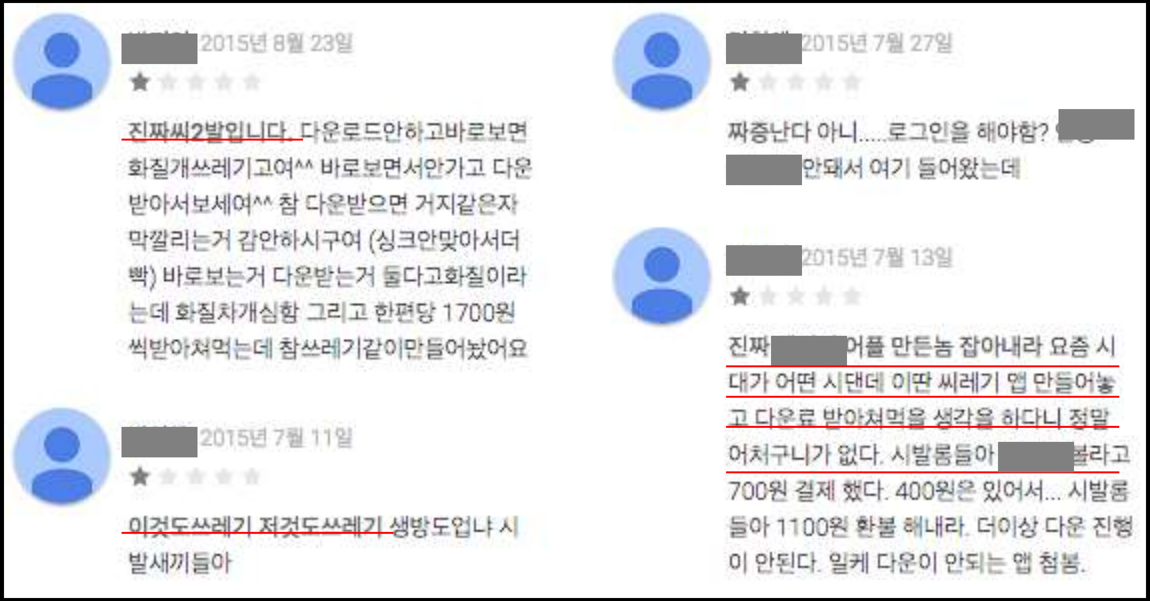
남들이 쓰지 않는 API라면?

RESTful 하지 않아도 된다. 사실 아무렇게나 만들어도 상관없다.
나만 이해하면 되니까... 그런데 바꾸어 말하면, 이런 API는 아무도 안쓴다는 얘기.

```
// RPC style request. What API client faces
http://fakesvc.com/getClient.php?name=chris

// API Server
<?php
...
$name = $_GET['name'];
$result = $conn->query("SELECT * FROM clients WHERE name='{$name}'");
while ($row = $result->fetch_object()) $payload[] = $row;
header('Content-Type: application/json; charset=utf-8');
echo json_encode($payload);
```

대충 개발하면 이렇게 된다!

What the fxxx…
오늘의 고객을 만족시키면 내일을 위한 영업은 필요 없다.



[redacted] 2015년 8월 23일
★☆☆☆☆
진짜씨2발입니다. 다운로드안하고바로보면 화질개쓰레기고여^^ 바로보면서안가고 다운받아서보세여^^ 참 다운받으면 거지같은자막깔리는거 감안하시구여 (싱크안맞아서더빡) 바로보는거 다운받는거 둘다고화질이라는데 화질차개심함 그리고 한편당 1700원씩받아쳐먹는데 참쓰레기같이만들어놨어요

[redacted] 2015년 7월 11일
★☆☆☆☆
이것도쓰레기 저것도쓰레기 생방도업나 시발새끼들아

[redacted] 2015년 7월 27일
★☆☆☆☆
짜증난다 아니.....로그인을 해야함? [redacted] 안돼서 여기 들어왔는데

[redacted] 2015년 7월 13일
★☆☆☆☆
진짜 [redacted] 어플 만든놈 잡아내라 요즘 시대가 어떤 시댄데 이딴 씨레기 앱 만들어놓고 다운료 받아쳐먹을 생각을 하다니 정말 어처구니가 없다. 시발롬들아 [redacted] 볼라고 700원 결제 했다. 400원은 있어서... 시발롬들아 1100원 환불 해내라. 더이상 다운 진행이 안된다. 일케 다운이 안되는 앱 첨봄.

오늘 이후, 우리는

1. REST 원칙을 **(대략)** 이해한다.
2. 남의 API를 **(대략)** 읽을 수 있다.
3. RESTful API를 **(대략)** 만들 수 있다.

그런데, API의 본질은 무엇일까요?

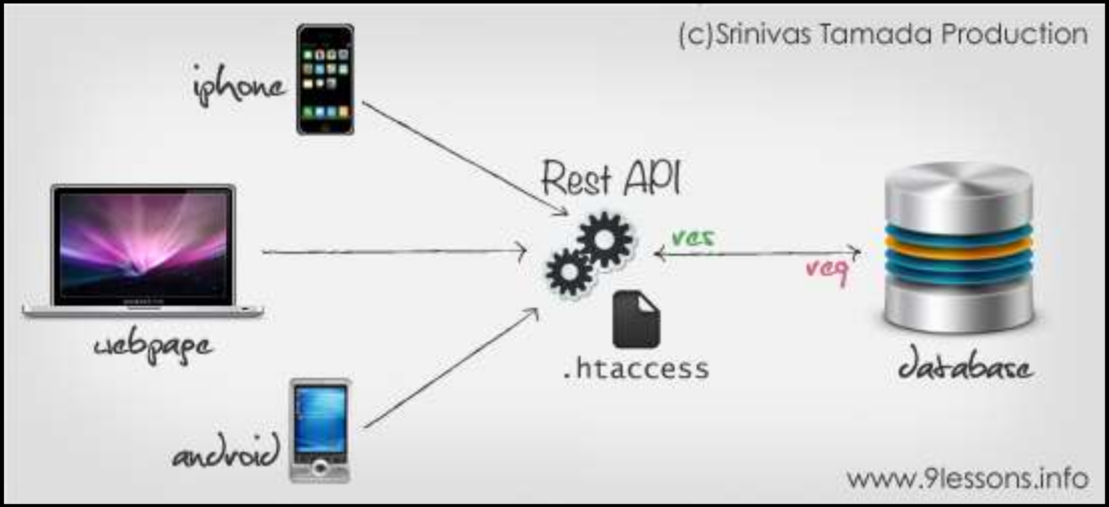
Separate of Concern

Decoupling

그럼, Web API의 본질은 무엇일까요?

Know who the API clients are. So various platforms, various languages

Decoupling + Platform Agnostic



REST Fundamental

REST의 역사

Representational State Transfer

웹(HTTP)의 창시자 중의 한사람인
Roy Fielding의 2000년 논문에서 소개

"현재 웹 서비스들의 HTTP의 본래 의도 및
우수성을 **제대로** 활용하지 못하고 있다"고 판단

웹의 장점을 최대한 활용할 수 있는
네트워크 기반의 아키텍처로 REST를 제안

REST의 특징

Uniform Interface (=Platform Agnostic)

Stateless

Cacheable

Self Descriptiveness

Client-Server Architecture

WHAT IS GOOD API? WHY REST?

Easy to learn, Easy to use,
Hard to misuse

+

Performance, Scalability, Reusability, Evolvability, Documentation

IS REST REALLY EASY TO IMPLEMENTERS?

REST is &*#!(@$^& Hard

REST <u>can</u> be easy

(if you follow some guidelines)

HOW TO LEARN REST

RESTful API Design Principles

(by Roy Fielding, Leonard Richardson, Martin Fowler, HTTP specification)

Web Giants 복붙

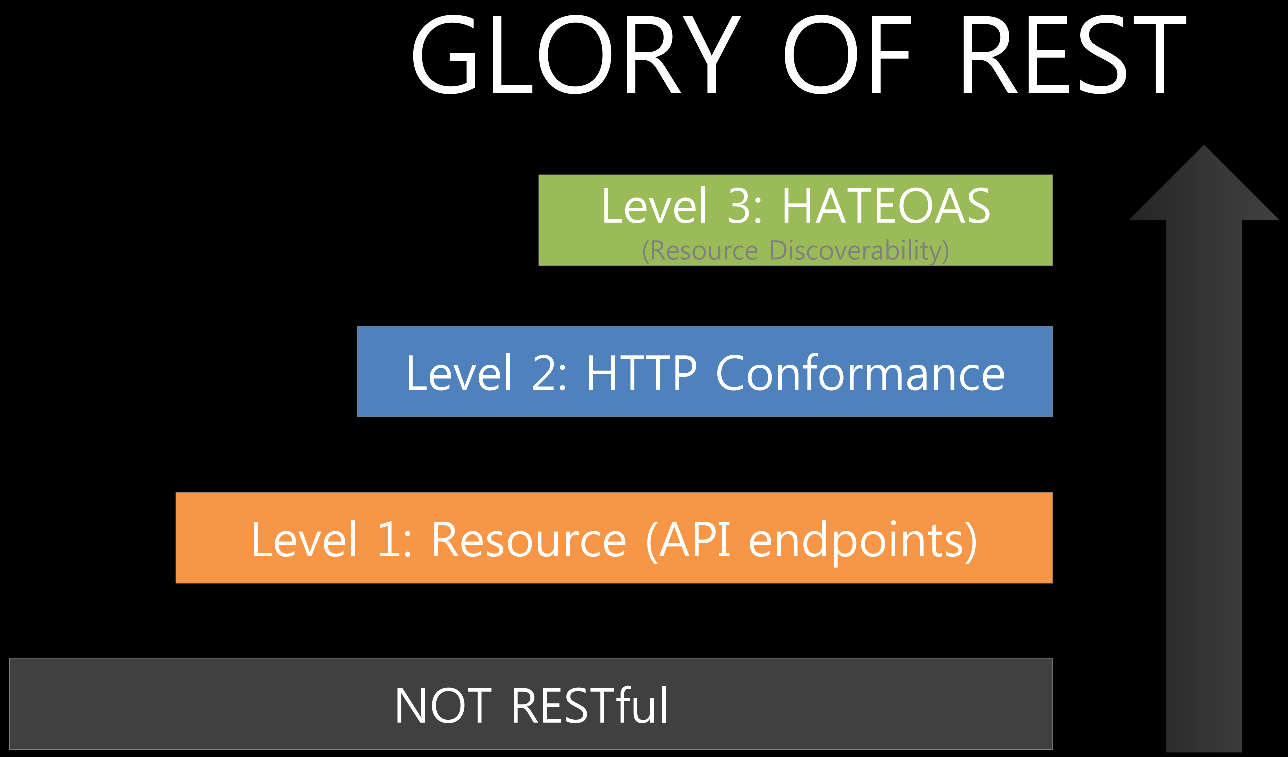
(Github, Twitter, Google, Facebook, …)

IT’S JUST A GUIDELINE! NOT A SPEC!

REST is an architectural style

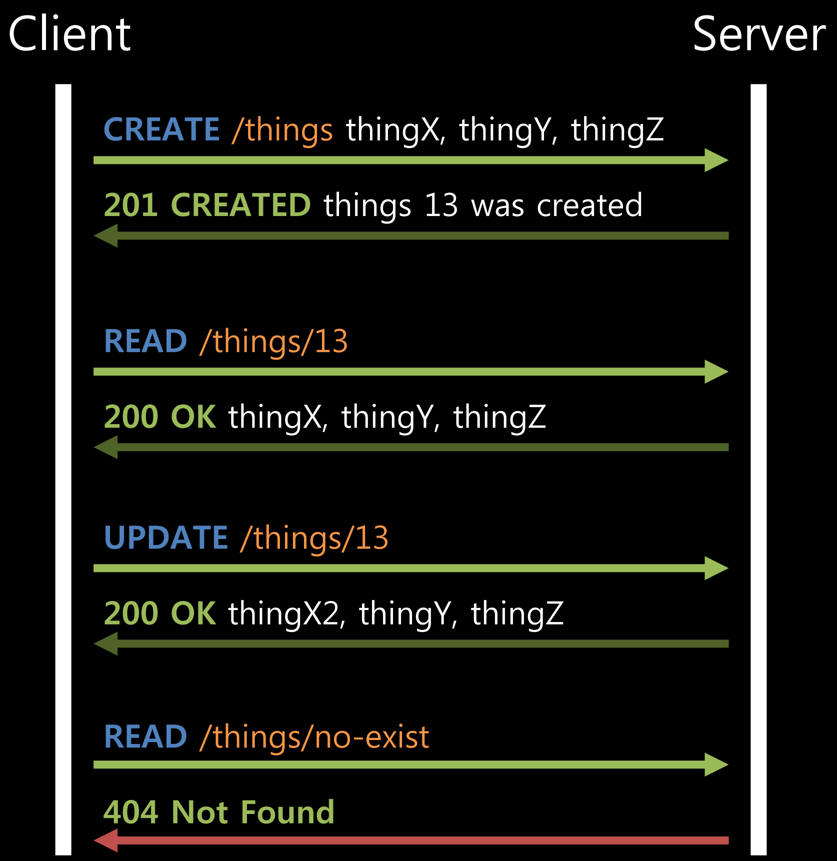
NO strict rule (~ful)

RICHARDSON MATURITY MODEL


GLORY OF REST
Level 3: HATEOAS
(Resource Discoverability)
Level 2: HTTP Conformance
Level 1: Resource (API endpoints)
NOT RESTful

http://martinfowler.com/articles/richardsonMaturityModel.html

HOW THE CLIENT & SERVER COMMUNICATE


Client
Server
CREATE /things thingX, thingY, thingZ
201 CREATED things 13 was created
READ /things/13
200 OK thingX, thingY, thingZ
UPDATE /things/13
200 OK thingX2, thingY, thingZ
READ /things/no-exist
404 Not Found

Notice that pattern (**COLOR**): A set of **commands (method)** performed on **things (resource)** generates **responses (message).**

This is the foundation of a REST API.

REpresentational **S**tate **T**ransfer

REST 101

3 Pillars of REST

Method, Resource, Message

POST

/things

201 CREATED
{"name": "teapot"}

METHOD

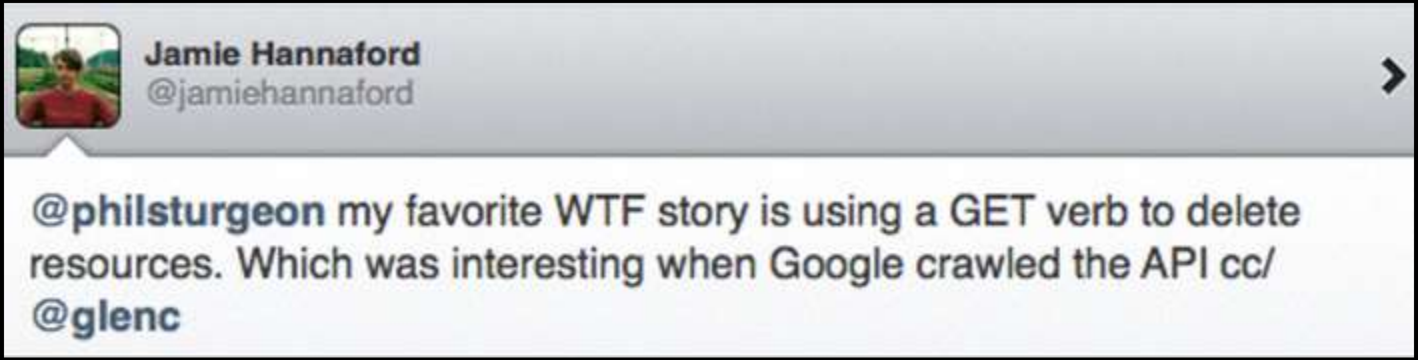
CRUD – Create, Retrieve(Read), Update, Delete

HTTP Method	Action	Endpoints	Controller Method (in Laravel)	Description	Idem-
POST	*Create*	*/things*	*store()*	*Create a new resource*	*No*
GET/HEAD	*Read*	*/things*	*index()*	*Get a collection of resource*	*Yes*
GET/HEAD	*Read*	*/things/a1b2*	*show($id)*	*Get the specified specified resource*	*Yes*
PUT/PATCH	*Update*	*/things/a1b2*	*update($id)*	*Update the specified resource*	*Yes*
DELETE	*Delete*	*/things/a1b2*	*delete($id)*	*Delete the specified resource*	*Yes*

METHOD - ANTI-PATTERN

Tunneling though GET or POST

```
GET http://fakehost/things?method=update&name=teapot          (X)

POST http://fakehost/things                                    (X)
{
  "getUser" : {"id": "teapot"}
}
```



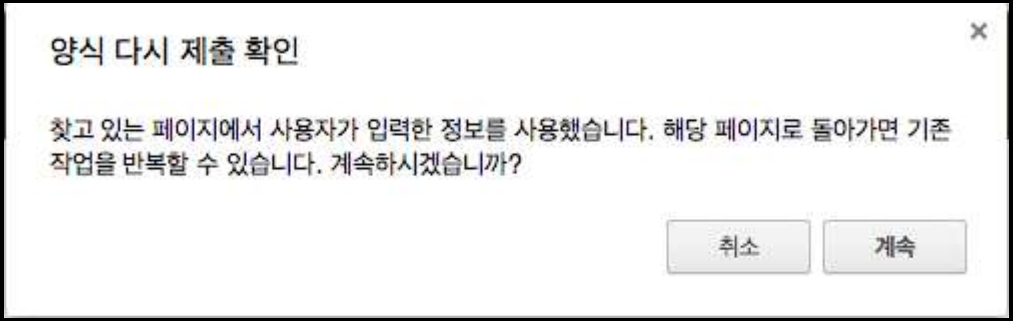
Jamie Hannaford
@jamiehannaford

@philsturgeon my favorite WTF story is using a GET verb to delete resources. Which was interesting when Google crawled the API cc/ @glenc

https://speakerdeck.com/philsturgeon/api-pain-points-lone-star-php-2015

METHOD - IDEMPOTENT

It's about SAFENESS when a request is repeated.
$a = 4; 와 $a++;의 차이. 실패할 경우, Rollback 해야 한다.



양식 다시 제출 확인

찾고 있는 페이지에서 사용자가 입력한 정보를 사용했습니다. 해당 페이지로 돌아가면 기존 작업을 반복할 수 있습니다. 계속하시겠습니까?

RESOURCE - NOUN

API Endpoint는 동사형(Verb) 사용을 지양하고, 명사형(Noun)을 사용한다.

```
// RPC style
GET http://fakehost/getThings                                   (X)

// RESTful style
GET http://fakehost/things                                      (O)

// Sub Resource
GET http://fakehost/author/{username}/things                    (O)
```

RESOURCE - PLURALIZE

API Endpoint는 단수(Singular) 보다 복수형(Plural)을 사용한다.

```
GET http://fakehost/thing                                      (X)

// Collection endpoint
GET http://fakehost/things                                     (O)

// Instance endpoint
GET http://fakehost/things/{id}                                (O)
```

RESOURCE – CONSISTENT CASE

API Endpoint의 대소문자를 일관되게 사용한다.

```
// Google, Facebook, Twitter, Github
GET http://fakehost/snake_case

// Google, Paypal
GET http://fakehost/spinal-case

// Google
GET http://fakehost/camelCase
```

RESOURCE - VERSIONING

새로운 Version을 위한 Controller로 Response 분리
또는 Reverse Proxy를 이용하여 서버를 물리적으로 분리

```
// Sub domain을 쓸 수 있다면
GET http://api.fakehost/v1/things


// Sub domain을 쓸 수 없는 경우
GET http://fakehost/api/v1/things
```

RESOURCE – DOMAIN NAME

대형 서비스라면… API Gateway, Developer Connection, and Authentication

```
// API(=Resource) Server
http://api.{company}.com

// Developer Portal
http://developers.{company}.com

// Authentication Server
http://oauth2.{company}.com
```

RESOURCE – ANTI-PATTERN

NOT self-descriptive
Method와 Endpoint만 보고도 무엇을 하는 것인지 이해할 수 있어야 한다.

```
// Request
POST http://nailshop/appointmentService
<openSlotRequest date="2015-09-30" nailist="gwen"/>

// Response
HTTP/1.1 200 OK
<openSlotList>
  <slot start="1400" end="1420"> … </slot>
</openSlotList>
```

RESOURCE – QUERY STRING

Complex things behind the ‘?’.
Pagination, Filtering, Sorting, Searching, Partial Response

```
// Pagination
/v1/things?range=0-25
/v1/things?offset=0&limit=25

// Filtering
/v1/things?origin=jp,en&rating=4,5

// Sorting
/v1/things?sort=name&direction=asc

// Searching
/v1/things?q=lorem+dolar
/v1/?q=lorem+dolar

// Partial Response
/v1/things?fields=title,created_at
```

REQUEST & RESPONSE – ANTI-PATTERN

all(); is bad – by Jeffrey Way

```php
<?php

class ThingsController extends Controller
{
    public function index()
    {
        return \App\Thing::all();
    }
}
```

REQUEST & RESPONSE – ANTI-PATTERN

all(); is bad – by Jeffrey Way

1. All is bad (Pagination)
2. No way to attach meta data
3. Linking DB structure to the API output
   (Hiding some field, Renaming field)
4. No way to signal Headers/response codes
   (Error response, HATEOAS)

REQUEST & RESPONSE – ANTI-PATTERN

Response Code (=Status Code)는 200만 있는게 아니다.

```
// AngularJS
$http.post("http://fakehost/things", {"name": "teapot"})
    .then(function(response) {
    if (response.status !== 200) {
        throw new Error("200이 아니면 죽음을 달라");
    }
});
```

REQUEST & RESPONSE – RESPONSE CODE

Code	String	Description
~~*1xx*~~		~~*All about information (Not used for API resp.)*~~
200	*OK*	*All about success*
201	*Created*	*Resource was created (Location Header)*
204	*No Content*	*Success response without response body*
3xx		*All about redirection*
304	*Not Modified*	*If-Modified-Since 에 대한 응답 (Cache)*
400	*Bad Request*	*All about client error*
401	*Unauthorized*	*Really means UNAUTHENTICATED*
403	*Forbidden*	*Really means UNAUTHORIZED*
404	*Not Found*	
405	*Method Not Allowed*	
422	*Unprocessable Entity*	*Validation Error (406 is also acceptable)*
500	*Server Error*	

REQUEST & RESPONSE – CONTENT NEGOTIATION

Content & Language Negotiation

```
// Request
GET /v1/things HTTP/1.1
Accept: application/json
Accept-Language: ko-KR;q=0.8,en-US;q=0.6,en;q=0.4

// Response
HTTP/1.1 200 OK
Content-Type: application/json
```

REQUEST & RESPONSE - TRAMSFORMATION

Transformer to decouple a PRESENTATION layer from the DOMAIN layer

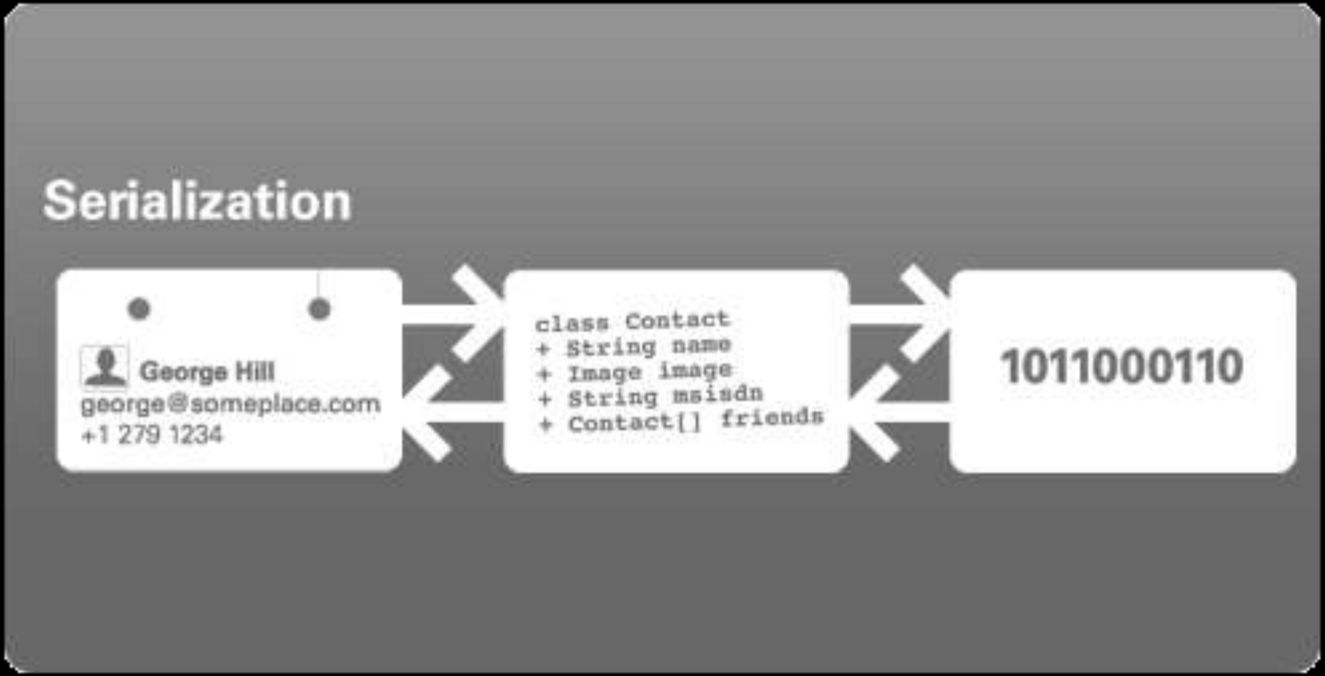
REQUEST & RESPONSE - TRANSFORMATION

Transformer to decouple PRESENTATION layer from DOMAIN layer

```php
<?php

class ThingsController extends Controller
{
    public function index()
    {
        return array_map(function($thing) {
            return [
                'id'   => (int) $thing->id,
                'meta' => ['version' => 1],
                '...'  => '...'
            ];
        }, \App\Thing::all());
    }
}
```

REQUEST & RESPONSE - SERIALIZATION

Serialize the response data to format it properly



```
class Contact
+ String name
+ Image image
+ String msisdn
+ Contact[] friends
```

REQUEST & RESPONSE – SERIALIZATION

league/fractal natively supports DataArraySerializer, ArraySerializer, JsonApiSerializer

```
// DataArraySerializer
{"data":{"foo":"bar"}} // instance
{“data":[{"foo":"bar"}]} // collection

// ArraySerializer
{"foo":"bar"} // instance
{"data":{"foo":"bar"}} // collection

// JsonApiSerializer
{"data":{"type":"books","id":1,"attributes":{"foo":"bar"}}} // instance
{"data":[{"type":"books","id":1,"attributes":{"foo":"bar"}}]} //
collection
```

http://fractal.thephpleague.com/serializers/

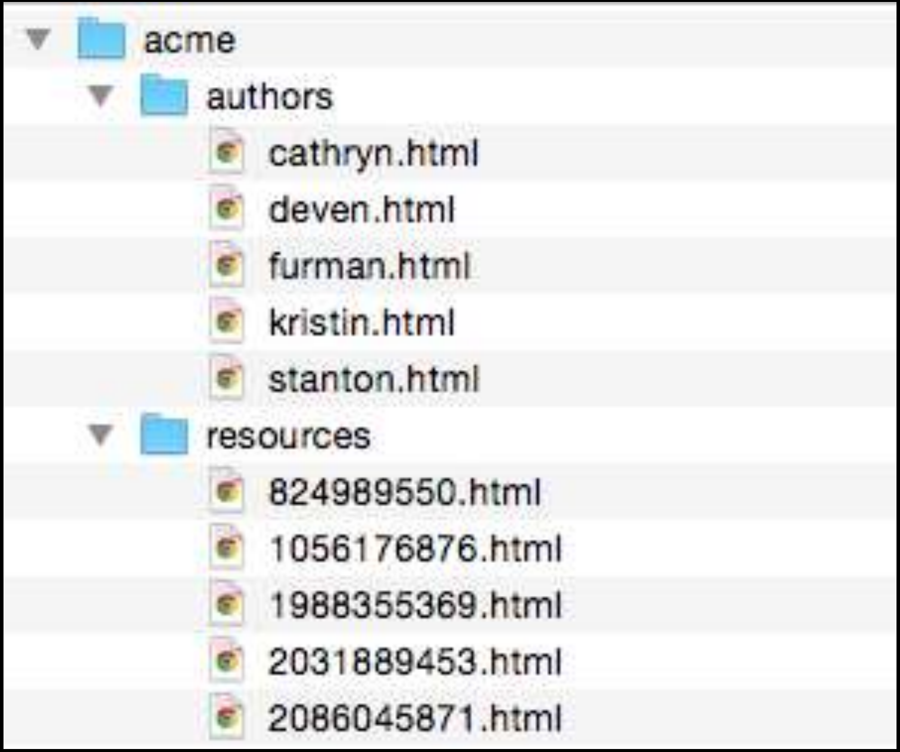
REQUEST & RESPONSE – ERROR RESPONSE BODY

Error response 는 최대한 Descriptive 하게.
DEBUG 모드에서는 Stack Trace등 디버그 정보도 출력하면 편리함.

```
// Response
{
  "error": {
    "code": 400,
    "message": "Deliberate Error",
    "dictionary": "http: //developers.fakehost/v1/errors/400"
  },
  "debug": {
    "line": 42,
    "file": "..."
  }
}
```

REQUEST & RESPONSE - HATEOAS

Hypermedia As The Engine Of Application State. In case of normal HTML response, client can navigate sub resources through set of links.



```
▼ acme
    ▼ authors
        cathryn.html
        deven.html
        furman.html
        kristin.html
        stanton.html
    ▼ resources
        824989550.html
        1056176876.html
        1988355369.html
        2031889453.html
        2086045871.html
```

REQUEST & RESPONSE - HATEOAS

API 사용자가 길을 잃지 않도록 Sub Resource(Nested)에 대한 링크를 포함.

```
// Response
{
  "data": [
    {
      "id": 100,
      "title": "Quia sunt culpa numquam blanditiis...",
      "link": {
        "rel": "self",
        "href": "http://fakehost/v1/things/100"
      }
    },
    {"...": "..."}
  ]
}
```

appkr/fractal

league/fractal Wrapper for Laravel5/Lumen5
Focuses on RESPONSE(=View layer) of the API response

HOW TO INSTALL

```
// composer.json
"require": {
  "appkr/fractal": "0.5.*",
  "league/fractal": "@dev",
}

$ composer update

// config/app.php
'providers'=> [
    Appkr\Fractal\ApiServiceProvider::class,
]
```

HOW TO USE

```php
// app/Http/routes.php
Route::resource(
    'things',
    ThingsController::class,
    ['except' => ['create', 'edit']]
);

// app/Http/Controllers/ThingsController
class ThingsController extends Controller
{
    public function index(\Appkr\Fractal\Http\Response $response)
    {
        return $response->setMeta(['version' => 1])->withPagination(
            \App\Thing::latest()->paginate(25),
            new \App\Transformers\ThingTransformer
        );
    }
}
```

HOW TO USE

```
// Response
{
  "data": [
    {
      "id": 1,
      "title": "Quia sunt culpa numquam blanditiis.",
      "created_at": "2015-09-19T08:07:55+0000",
      "link": {
        "rel": "self",
        "href": "http://localhost:8000/v1/things/1?include=author"
      },
      "author": "landen08"
    },
    {
      "...": "..."
    }
  ],
  "meta": {
    "version": 1,
    "pagination": {
      "total": 106,
      "count": 25,
      "per_page": 25,
      "current_page": 1,
      "total_pages": 5,
      "links": {
        "next": "http://localhost:8000/v1/things/?page=2"
      }
    }
  }
}
```

HOW TO ACTIVATE & TEST EXAMPLES

Database migrations and seeder, Routes definition, Eloquent Model and corresponding Controller, FormRequest, Transformer, Integration Test

```
// Activate examples at vendor/appkr/fractal/src/ApiServiceProvider.php
$this->publishExamples();


// Migrate/seed tables at a console
$ php artisan migrate --path="vendor/appkr/fractal/database/migrations"
$ php artisan db:seed —class="Appkr\Fractal\Example\DatabaseSeeder"

// Boot up a local server
$ php artisan serve

// Request to the endpoint using CURL or POSTMAN
GET http://localhost:8000/v1/things
GET http://localhost:8000/v1/things/{id}
POST http://localhost:8000/v1/things
PUT http://localhost:8000/v1/things/{id} // Method overriding required
DELETE http://localhost:8000/v1/things/{id} // Method overriding
required
```

떡밥!!!
스스로 찾아 더 공부해 보세요

AUTO-INCREMENT ID (BAD DESIGN)

zackkitzmiller/timy, jenssengers/optimus

인증과 권한부여가 없는 식당예약 API를 호출했더니,
내 식당 예약이 /reservations/15로 리턴됨.

"덕질"
/reservations/14 요청해 보실거죠?

여기까지.. 더하면 잡혀 가요.

AUTO-INCREMENT ID (BAD DESIGN)

구현 참고 http://laravel.io/bin/JxXyW

```
// app/Providers/AppServiceProviders.php
public function register() {
    $this->app->singleton(Optimus::class, function () {
        return new \Jenssegers\Optimus(1118129599, 664904255, 1002004882);
    });
}

// app/BaseModel.php
abstract class BaseModel extends Model {
    public function getIdAttribute($value) {
        return app(Optimus::class)->encode($value);
    }

    public static function find($id, $columns = ['*']) {
        $transformedId = app(\Jenssegers\Optimus\Optimus::class)->decode($id);
        return static::query()->find($transformedId, $columns);
    }
}

// app/SomeModel.php
class SomeModel extends BaseModel { }

// app/Http/Controllers/SomeController.php
public function show($id) {
    return SomeModel::find($id);
}
```

AUTHENTICATION

`laravel/socialite, league/oauth2-server, tymon/jwt-auth`

Access Control, Throttling, …
등을 위해 필요합니다.

1st/3rd Party Oauth2, JWT 이용

SSL 권장

AUTHENTICATION

3rd Party Auth/JWT 구현 참고 appkr/rest

```
// app/Http/Controllers/Api/Auth/SocialController.php
protected function handleProviderCallback($provider) {
    $user = Socialite::with($provider)->user();

    return $user = $this->repo->firstOrCreate([
        '...',
        'email' => $user->getEmail(),
        'avatar' => $user->getAvatar()
    ], $this);
}

// app/Http/Controllers/Api/Auth/SessionController.php
protected function respondLoginSuccess(AuthRequest $request, User $user) {
    $meta = $this->buildUserMeta($user);
    return $this->response->setMeta(['token' => \JWTAuth::fromUser($user)])->success();
}

// app/Http/Controllers/Api/ThingsController.php
public function __construct() {
    $this->middleware('jwt.auth', ['except' => ['index', 'show']]);
}

// Request
POST /v1/things
Authorization: Bearer header.payload.signature
```

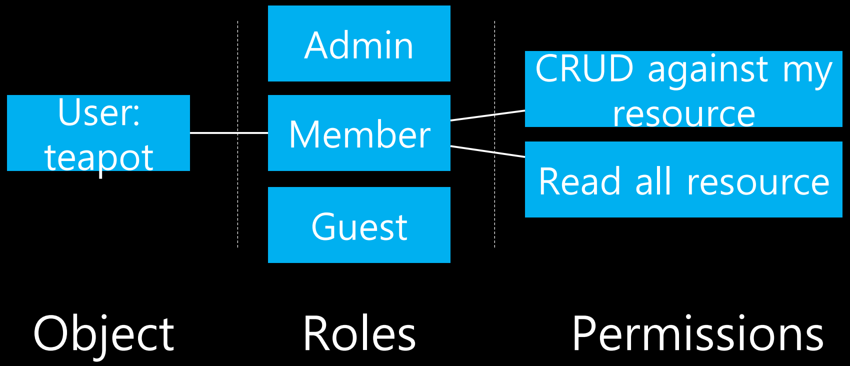
AUTHORIZATION

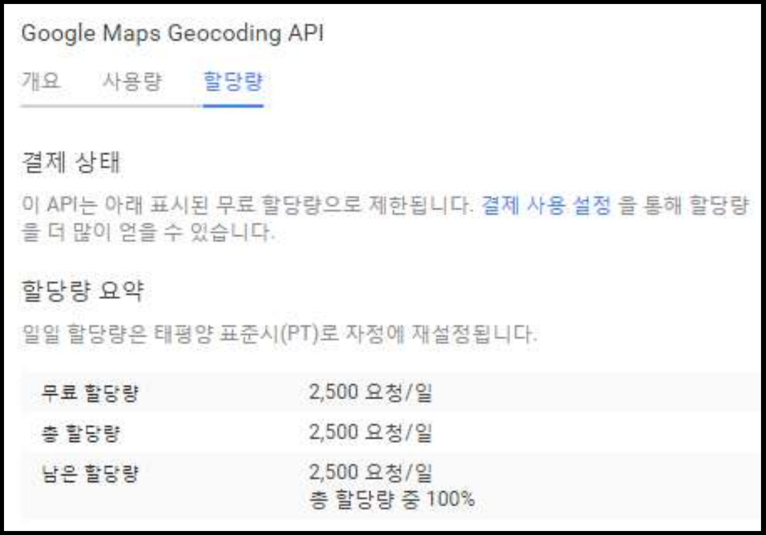
bican/roles, cartalyst/sentinel, graham-campbell/throttle

RBAC + Laravel Native

API의 공평한 분배를 통한 정의사회 구현

Data가 재산인 시대에 Data API로 돈 벌려면 Throttle은 필수




Google Maps Geocoding API

개요 사용량 할당량

결제 상태

이 API는 아래 표시된 무료 할당량으로 제한됩니다. 결제 사용 설정 을 통해 할당량을 더 많이 얻을 수 있습니다.

할당량 요약

일일 할당량은 태평양 표준시(PT)로 자정에 재설정됩니다.

무료 할당량	2,500 요청/일
총 할당량	2,500 요청/일
남은 할당량	2,500 요청/일 총 할당량 중 100%

http://bcho.tistory.com/955

AUTHORIZATION

RBAC, Throttle 구현 참고 appkr/rest

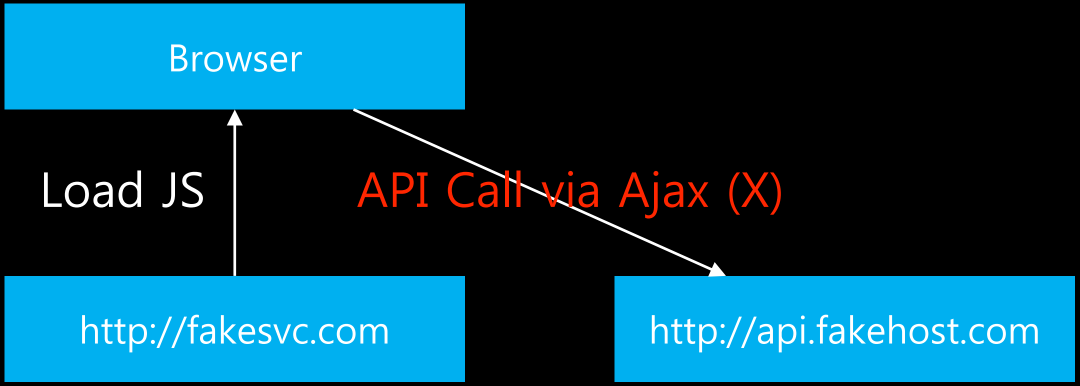
```php
// app/Http/Middleware/RoleMiddleware.php
public function handle($request, Closure $next, $role) {
    if (! $request->user()->is($role)) {
        if ($request->ajax() or $request->is('v1/*')) {
            throw new Exception('Forbidden', 403);
        }
    }

    return $next($request);
}

// app/Http/Middleware/ThrottleMiddleware.php
public function handle($request, Closure $limit = 10, $time = 60) {
    if (! $this->throttle->attempt($request, $limit, $time)) {
        throw new TooManyRequestHttpException($time * 60, 'Rate limit exceeded.');
    }

    return $next($request);
}

// app/Http/Controllers/Api/ThingsController.php
public function __construct() {
    $this->middleware('role:member', ['except' => ['index', 'show']]);
    $this->middleware('throttle:10,1');
}
```

CORS

Reverse Proxy가 최선. 차선은 Access-Control-Allow-Origin HTTP Header 조작: Barryvdh/laravel-cors, asm89/stack-cors. 폭탄 넘기기: jsonp

API 서버로 Ajax 요청을 하는 JavaScript를 실행하는 도메인과, API 서버의 도메인이 다르면 에러납니다. Same Origin Policy

Scheme, Port도 같아야 해요!!!



CACHING

Application(Domain) Level의 Cache
Cache에 적재해 놓고, Create, Update, Delete 액션이 발생하면 Cache 삭제

```
// app/Http/Controllers/v1/ThingsController.php
class ThingsController extends Controller
{
    public function index()
    {
        \Cache::get('things', function () {
            return \App\Things::all();
        });
    }


    public function store()
    {
        ...
        event(new \App\Events\ClearCache('things'));
    }
}
```

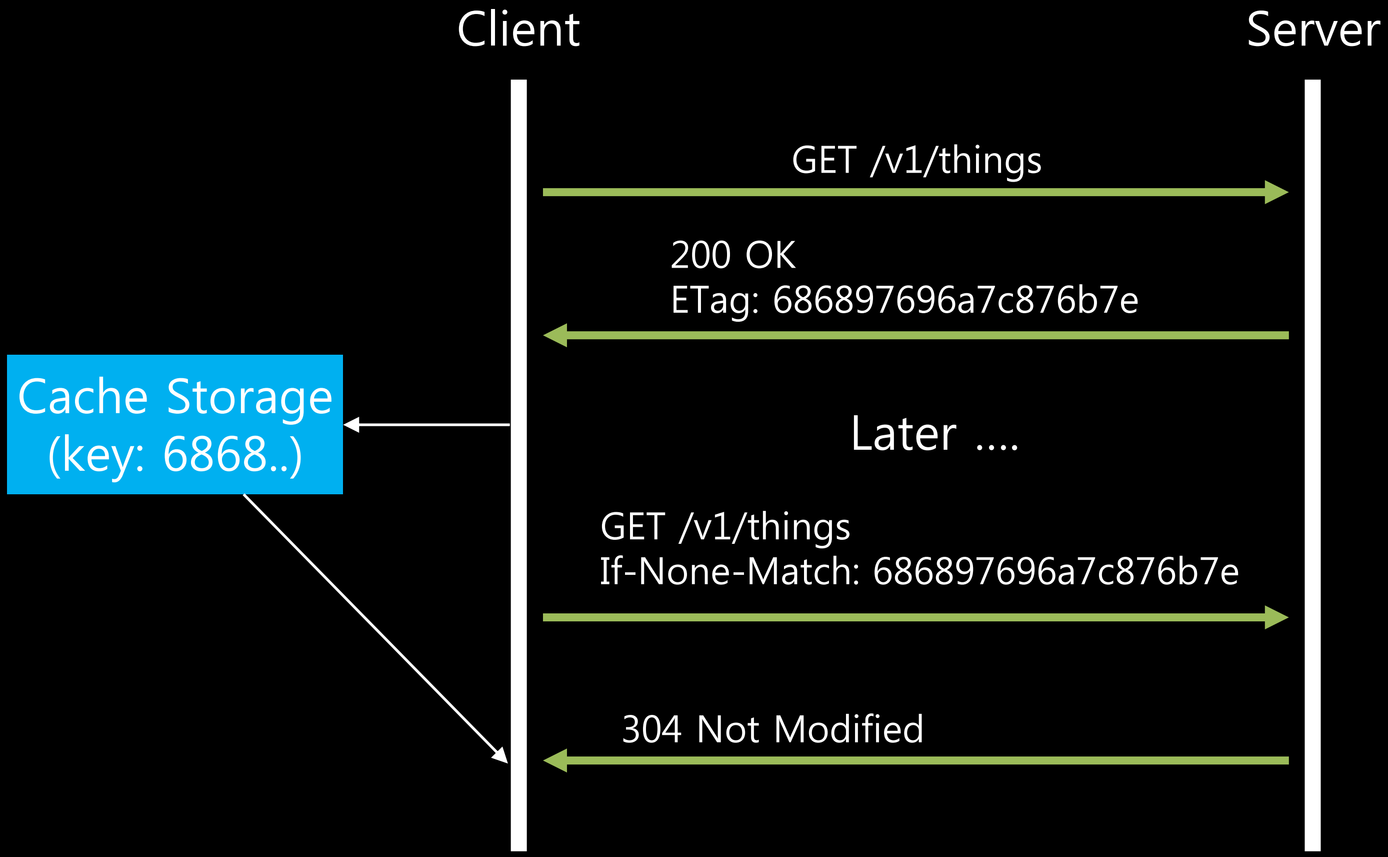
CACHING

Application(Domain) Level의 Cache
Cache에 적재해 놓고, Create, Update, Delete 액션이 발생하면 Cache 삭제

```php
// app/Events/ClearCache.php
public function __construct($tags)
{
    $this->tags = $tags;
}

// app/Listeners/CacheListener.php
public function handle(ClearCache $event)
{
    if (!is_array($event->tags)) {
        return $this->cache->tags($event->tags)->flush();
    }



    foreach ($event->tags as $tag) {
        $this->cache->tags($tag)->flush();
    }

    return;
}
```

CACHING

ETAG/If-None-Match 또는 Last-Modified/If-Modified-Since 를 이용한 Cache.
API 소비자는 Web Browser만 있는게 아니다.



https://en.wikipedia.org/wiki/HTTP_ETag

API MODELING LANGUAGE/SERVICE

http://raml.org/, https://apiblueprint.org/, https://apiary.io/

REST API Modeling Language
(혼자 개발하기에) 저는 안 써 봤습니다;;;
(팀웍하신다면) 한번 써 보세요.

FURTHER READINGS

RESTful API에 대해 더 공부하고 싶거나,
Laravel에서 API 구현 방안에 대해 더 공부하고 싶으시다면..

- http://www.slideshare.net/landlessness/teach-a-dog-to-rest, https://www.youtube.com/watch?v=QpAhXa12xvU
- https://laracasts.com/series/incremental-api-development

Summary

1. REST 원칙을 이해한다.
   Method, Resource, Message(Response)

2. 남의 것을 읽을 수 있다.
   API Document를 모두 읽을 필요는 없다.

3. 내 것을 만들 수 있다.
   직접, appkr/fractal, dingo/api 이용.

Email: juwonkim@me.com

Facebook: juwonkimatmedotcom

Github: appkr