

ㅋ 이 만화 보셨죠?

<http://emptydream.tistory.com/m/post/3852>

태어날 때부터 아는 사람이 어디 있겠습니까?  
모르면 배우면 됩니다.

"배움은 모르는 것이 많다"는  
사실을 깨닫는 과정이다.  
제가 자주 쓰는 말입니다.

# REST는 PUG에서도 Hot한 주제

The screenshot shows a Slack interface with a sidebar on the left and a main chat area on the right. The sidebar lists channels like #general and #oss, and direct messages. The main chat area shows a conversation in the #general channel on June 9th, 2016. The search bar at the top right contains the text 'REST'. The chat history shows a message from kangseunguk asking about REST API design, followed by a response from appkr and another from kangseunguk.

Slack

#general 186 members

Q REST

June 9th

**kangseunguk** 6:30 PM  
REST API 설계에서  
GET->LIST  
POST->INSERT  
PUT->UPSERT  
DELETE->REMOVE라고 했을때 로그인같이 민감한 부분이 기존에  
POST로 되어있는데  
그냥 예외적으로 로그인 API는 POST를 다른의미로 써야하는건가요?  
머리아프네요

**appkr** 6:33 PM  
로그인은 세션 create 의 의미 입니다.  
👍 1

**kangseunguk** 6:33 PM  
아 세션 만든다  
오호

Viewing archives from June 9th, 2016 - June ... Jump to recent messages ↓

+ You are viewing the archives of #general 😊

# REST는 PUG에서도 Hot한 주제

The screenshot shows a Slack interface with a sidebar on the left and a main chat area on the right. The sidebar displays the workspace name 'Modern PHP Us...' and a list of channels including #general, #oss, #pragmatic\_medernizing, #qna, and #random. The main chat area shows a conversation in the #general channel dated July 27th. The conversation starts with a message from munkyo.seo at 8:55 PM, followed by a response from appkr at 8:57 PM, and another message from munkyo.seo at 8:57 PM.

**#general** 186 members | Search | @ | ☆

July 27th

**munkyo.seo** 8:55 PM  
오늘 정말 억이고 싶지 않은 외주업체 개발자를 만났는데...  
저희 회사는 API통해서 외부와 연동하고 있고, API는 restful api 사용하고 있습니다.  
라고 말하니까  
“레스트풀이 뭐예요?” 라고 (개발자임)

**appkr** 8:57 PM  
허허 용감한 분이시네요. 제가 그 상황이라면 묻지않고 기억해뒀다 찾아봤을거예요.

**munkyo.seo** 8:57 PM  
POS 뒷단 서버를 만드는데...  
"포스 댓수는 대략 300개 뒷단 데이터 수집하는 서버에 필요한 서버 용량은 얼마면 될까요?"라고 여쭙보니  
한 2테라는 있어야합니다. 라고 ;;;  
어이가 없어서 왜인지 꼬치꼬치 캐물어봤더니  
모든 액션을 로그로 남기는데 그 로그를 DB에 모두 저장하더군요 ㄷ ㄷ

# RESTful URL 설계

2016년 8월 3일  
appkr(김주원)



**modern**  
php user group

# URL 설계가 중요한 이유

- 애플리케이션의 진입점  
즉, 로직 설계의 첫 걸음
- 검색 크롤러가 가장 처음 만나는  
것도 URL (=>SEO)
- 서버 플랫폼/구현 로직 디커플링
- 공개 후 변경 비용이 비싸다.

*Why?*

# 같은 플랫폼, 다른 느낌



[http://clien.net/cs2/bbs/board.php?  
bo\\_table=cm\\_mac&wr\\_id=791884](http://clien.net/cs2/bbs/board.php?bo_table=cm_mac&wr_id=791884)

VS



[https://en.wikipedia.org/wiki/  
Display\\_resolution](https://en.wikipedia.org/wiki/Display_resolution)

[https://en.wikipedia.org/w/index.php?  
title=Display\\_resolution](https://en.wikipedia.org/w/index.php?title=Display_resolution)

display resolution - Google x appkr

← → ↻ <https://www.google.co.kr/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=...> ☆ JB O V ABC S ≡

Google **display resolution** [Keyboard] [Voice Search] [Search]

전체 이미지 동영상 뉴스 지도 더보기 ▾ 검색 도구 [User] [Refresh] [Settings]

검색결과 약 268,000,000개 (0.50초)

관련검색: [display resolution table](#) [mobile display resolution](#) [lcd resolution](#) [video resolution](#)  
[display resolution chart](#)

**MacBook**  
[광고] [www.apple.com/kr](http://www.apple.com/kr) ▾  
시대를 가볍게 뛰어넘다. 더 알아보기.

**Display resolution - Wikipedia, the free encyclopedia**  
[https://en.wikipedia.org/wiki/Display\\_resolution](https://en.wikipedia.org/wiki/Display_resolution) ▾ 이 페이지 번역하기  
The display resolution or display modes of a digital television, computer monitor or display device is the number of distinct pixels in each dimension that can be ...

**List of common resolutions**  
Computer graphics[edit]. For the table below, ... A list of standard ...

**Graphics display resolution**  
The graphics display resolution is the width and height ...

**Computer display standard**  
Computer display standards are a ...  
Computer display standard.

**Display size**  
On 2D displays, such as computer monitors and TVs, the display ...

URL Rewrite 방법은 아래 링크 참고하세요.  
<http://blog.appkr.kr/work-n-play/importance-of-service-url/>



# URL 구조



피자 주문을 URL로 표현하면,

`ko://awesomepizza.samsung3.seoul:80/pizza/combo?dough=thin&tomato=0#piece-1`

# URL을 바라보는 관점의 차이

- **R**emote **P**rocedure **C**all

GET /getAllArticles

GET /getArticle?id=1

POST /saveDog

=> 원격 서버의 함수 호출. URL == API 함수명

- **R**Epresentational **S**tate **T**ransfer

GET /articles/{id}

POST /articles

=> 원격 서버의 리소스(데이터)에 대한 상태 교환

# REST

- 로이 필딩(Roy Fielding)  
"현재의 웹 서비스들이 HTTP의 본래 의도  
및 우수성을 제대로 활용하지 못하고 있다"
- 웹의 장점을 최대한 활용하는  
이종(異種, heterogeneous)  
시스템간의 네트워크 통신 구조



# REST

- 엄격하게 지켜야 하는 "표준"은 아니지만...
- 안 지키면 "x 팔린다"
- 왜? 이름만 대면 아는 인터넷 거물들은 이 원칙을 지킨다.
- RESTful, "REST 원칙을 따르는"

de facto

# REST 원칙의 구성 요소

## 1. 메서드 (method)

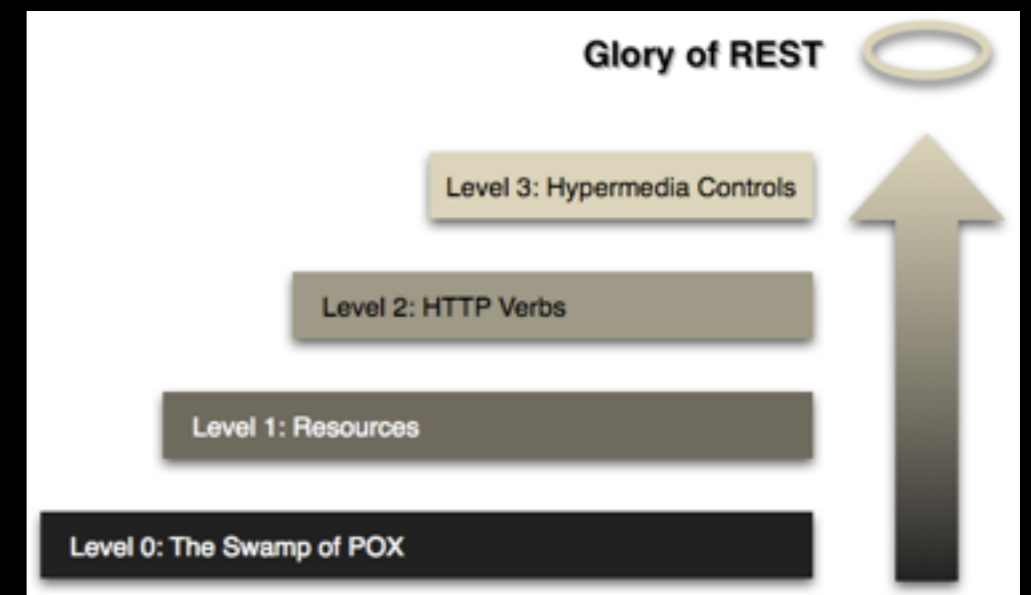
HEAD, GET, POST, PUT/PATCH, DELETE

## 2. 리소스 (resource)

Article, Comment

## 3. 메시지 (message)

HTTP 상태 코드 및 본문



Richardson Maturity Model

<http://restcookbook.com/Miscellaneous/richardsonmaturitymodel/>

# 10가지 모범 사례 (Best Practice)

# 1. 적절한 HTTP 메서드를 사용한다.

- 리소스의 상태를 읽을 때는 GET, 상태를 변경할 때는 POST, PUT/PATCH, DELETE 메서드를 사용한다.
- CRUD Mindsets

DELETE /articles/1

GET /deleteArticles?id=1 (X)



## 2. HTTP 메서드 오버라이드

- 일부 브라우저, 네트워크 프록시는 GET, POST만 쓸 수 있다.  
PUT(PATCH), DELETE 요청할 때는 서버에 힌트를 제공한다.

```
POST /articles
---payload---
_method=PUT&title=...&content=...
```

or

```
POST /articles
X-HTTP-Method-Override=PUT
---payload---
title=...&content=...
```



### 3. 리소스는 명사로 표현한다.

- 리소스는 서버에 저장된 데이터(모델)이다.  
URL 엔드포인트는 "컬렉션"과 "인스턴스" 딱 두가지 형태.

형태	리소스 (엔드포인트)	GET	POST	PUT /PATCH	DELETE
컬렉션	/articles	글 목록	글 저장	없음	글 전체 삭제
인스턴스	/articles/ {id}	{id} 글 조회	없음	{id} 글 수정	{id} 글 삭제

## 4. 복수형 리소스 이름, 일관된 대소문자

- 복수형 리소스 이름

```
GET /articles  
GET /article (X)
```

- URL뿐만 아니라, 필드 이름에도 일관된 대소문자 규칙 적용

```
GET /push_messages # 스네이크 표기법 적용  
{  
  "total": 1540,  
  "perPage": 10, # 낙타 표기법 적용  
  "current-page": 1, # 대시 표기법 적용 (주의)  
  "data": [...]  
}
```

## 5. 관계를 노출할 때는 리소스 중첩

{id}보다는 {slug}

GET /tags/{id}/articles

GET /tags/{id}?sub\_model=articles (X)

## 6. 컬렉션과 인스턴스 조회에서 복잡한 것들은 물음표(?) 뒤에서 표현한다.

```
GET /articles?q=Lorem # 검색
```

```
GET /articles?sort=view_count&order=asc  
# 정렬
```

```
GET /articles?page=2 # 페이징
```

```
GET /articles?fields=id,title  
# 필드 선택(Partial Response)
```

# 7. 알맞는 HTTP 응답 코드를 사용한다.

200 - Ok	# 성공
201 - Created	# 리소스 생성 성공
204 - No Content	# 리소스 삭제 성공 등에 주로 사용
304 - Not Modified	# 클라이언트에 캐시된 리소스 대비 서버 리소스의 변경이 없음
400 - Bad Request	# 클라이언트의 요청 오류
401 - Unauthorized	# 인증 필요 (실제로는 Unauthenticated 의미)
403 - Forbidden	# 권한 부족 (실제로는 Unauthorized 의미)
404 - Not Found	# 요청한 리소스가 없음

# 7. 알맞는 HTTP 응답 코드를 사용한다(계속).

- 405 – Method Not Allowed # 서버에 없는 URL 엔드포인트
- 406 – Not Acceptable # Accept\* 요청 헤더를 수용할 수 없음
- 409 – Conflict # 기존 리소스와 충돌
- 410 – Gone # 리소스가 삭제됨
- 422 – Unprocessable Entity # 유효성 검사 오류
- 429 – Too Many Requests # 사용량 초과 오류
- 500 – Internal Server Error # 서버에서 요청 처리 중 오류
- 503 – Service Unavailable # 서버가 일시적으로 응답할 수 없음

## 7. 알맞는 HTTP 응답 코드를 사용한다(계속).

- **이하 모든 내용은 API 개발에만 적용된다.**
- 응답 코드 뿐만아니라 클라이언트/개발자가 이해할 수 있는 응답 본문을 제시

POST /articles

```
{
  "errors": {
    "code": 429,
    "message": "too_many_requests"
  }
}
```

## 8. 길을 잃지 않도록 한다(API only).

- HATEOAS (Hypermedia as the Engine off Application State)
- HTML은 메뉴나 링크로 다른 페이지로 이동할 수 있다. 반면, 데이터 자체가 응답 메시지인 API에서는 서버에 어떤 다른 리소스가 있는지 모른다.



## 8. 클라이언트가 길을 잃지 않도록 한다.

GET /articles

```
{
  data: [
    {
      id: 1,
      title: "...",
      links: [
        rel: "self",
        href: "http://api.myapp.dev:8000/v1/articles"
      ]
      user: {
        id: 5,
        name: "...",
        links: [
          rel: "self",
          href: "http://api.myapp.dev:8000/v1/users/5"
        ]
      }
    },
    {"..."}
  ]
}
```

## 9. API 버전

GET `http://api.example.com/v1/articles`

GET `http://example.com/api/v1/articles`

GET `http://example.com/api/articles`

Accept: `application/vnd.example.v1`

GET `http://example.com/api/articles`

Accept: `application/vnd.example.article+json;  
version=1`

# 10. 콘텐츠 및 언어 협상

# Request

GET /articles

Accept: application/json

Accept-Language: ko-KR

-----

# Response

HTTP/1.1 200 OK

Content-type: application/json

```
{"message": "안녕하세요?"}
```

# 데모

라라벨 몰라도 느낌만...

```
1 +-----+-----+-----+-----+
2 | Method   | URI           | Name           | Action          | Middleware     |
3 +-----+-----+-----+-----+
4 | GET|HEAD  | articles     | articles.index | ArticlesController@index | web           |
5 | POST      | articles     | articles.store | ArticlesController@store | web,auth     |
6 | GET|HEAD  | articles/create | articles.create | ArticlesController@create | web,auth     |
7 | GET|HEAD  | articles/{id} | articles.show  | ArticlesController@show  | web           |
8 | PUT|PATCH| articles/{id} | articles.update | ArticlesController@update | web,auth     |
9 | DELETE    | articles/{id} | articles.destroy | ArticlesController@destroy | web,auth     |
10 | GET|HEAD  | articles/{id}/edit | articles.edit | ArticlesController@edit  | web,auth     |
11 +-----+-----+-----+-----+
```

동사 바꿀건 ;;

Runner Import Builder Team Library appkr IN SYNC No environment

Search

History Collections

All Me Team

MyApp 8 requests

GET http://api.myapp.dev:8000/v1

GET http://api.myapp.dev:8000/v1/articles

GET http://api.myapp.dev:8000/v1/articles?include=...

POST http://api.myapp.dev:8000/auth/register

POST http://api.myapp.dev:8000/auth/login

POST http://api.myapp.dev:8000/v1/articles

PUT http://api.myapp.dev:8000/v1/articles/2228821...

**DEL http://api.myapp.dev:8000/v1/articles/9466971...**

Postman Echo 21 requests

http://api.myapp.dev:8000/v1/articles/946697172

DELETE http://api.myapp.dev:8000/v1/articles/816856943 Params Send Save

Authorization Headers (3) Body Pre-request Script Tests Generate Code

Accept	application/json	
Content-Type	application/json	
Authorization	Bearer eyJ0eXAlOIJKV1QILCjh	
key	value	

Body Cookies Headers (11) Tests Status: 204 No Content Time: 188 ms

Pretty Raw Preview Save Response

# 요점 정리

# 나는 이제 000을 안다.

- URL 설계의 중요성
- URL의 의미
- RPC vs. REST의 차이점
- REST의 세가지 구성 요소
- 모범 사례 10가지
  1. HTTP 메서드
  2. 메서드 오버라이드
  3. 명사형, 컬렉션과 인스턴스
  4. 복수형, 대소문자 일관성
  5. 중첩된 리소스 표현
  6. 물음표를 이용한 조회
  7. HTTP 응답 코드 및 본문
  8. HATEOAS
  9. API 버저닝
  10. 콘텐츠 및 언어 협상



# "MUST" LIST

Github API는 REST 바이블

<https://developer.github.com/v3/>

# "SHOULD" LIST

## Teach a Dog REST

<http://www.slideshare.net/landlessness/teach-a-dog-to-rest>

## RESTful API Design - Second Edition

<https://www.youtube.com/watch?v=QpAhXa12xvU>

## API Pain Points

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

# 광고 1

오늘 사용한 예제 코드는

9월에 제이펍으로 출간 예정인

라라벨로 배우는 실전 웹 프로그래밍(가칭)

도서의 예제 코드입니다.

<https://github.com/appkr/l5code>

## 광고 2

쉬고 계신 분들  
한번 보고 가시면  
참 좋은데...

2016 메쉬코리아 공개 채용

네 당신이야.







Stay hungry. Stay foolish.

Steve Jobs

quote fancy