AGAIN2011 Workshop
Keynote #2

RESTful URL 설계

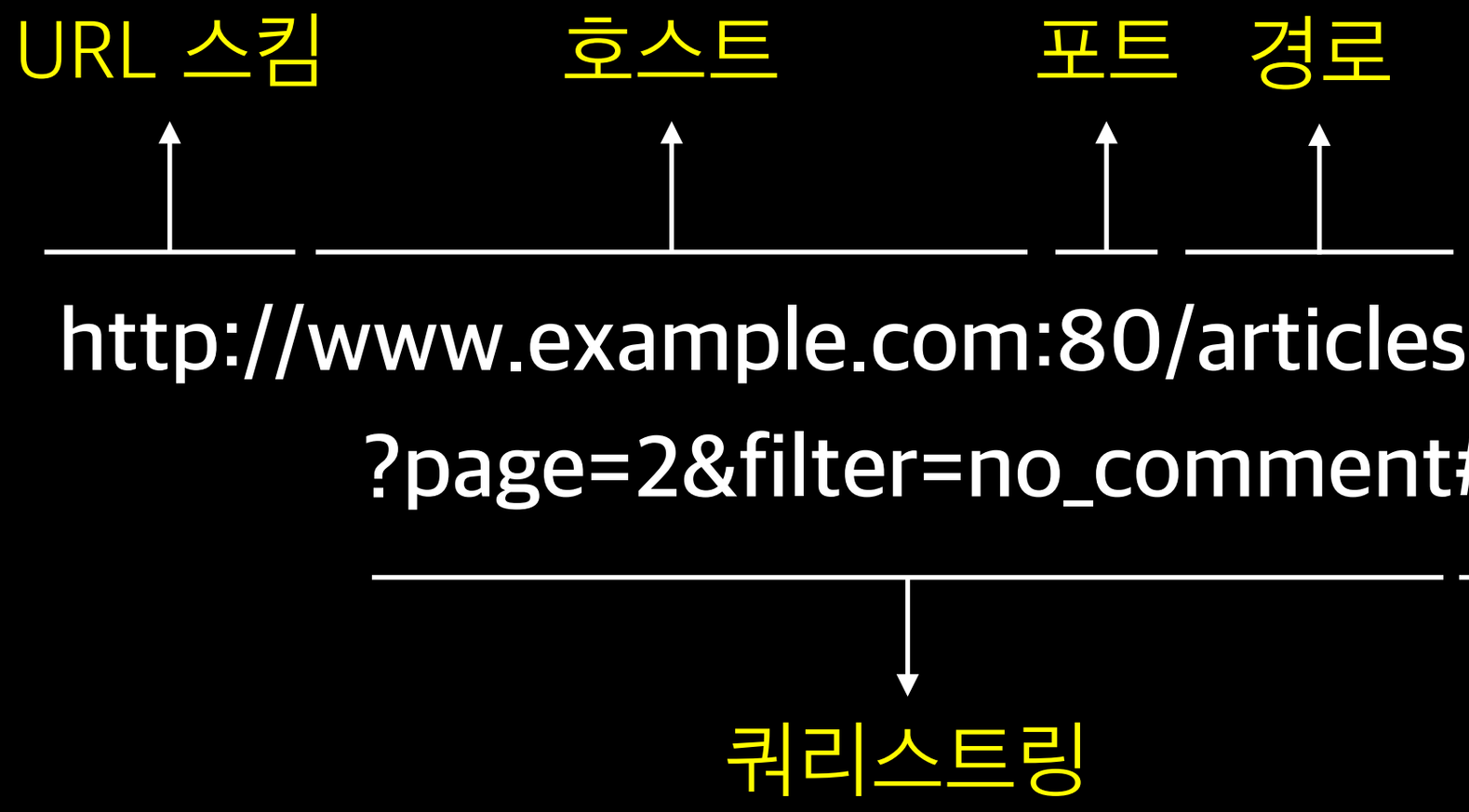
2016. 6. 26
appkr(R팀 멘토)

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.

서비스 URL의 중요성 및 변경의 비용 http://blog.appkr.kr/work-n-play/importance-of-service-url/#22-pretty-url

URL 구조



피자 주문을 URL로 표현하면, ko::://awesomepizza.samsung3.seoul:80/pizza/combination?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

- **RE**presentational **S**tate **T**ransfer

  ```
  GET /articles/:id
  POST /articles
  ```

  => 원격 서버의 리소스(데이터)에 대한 상태 교환

REST

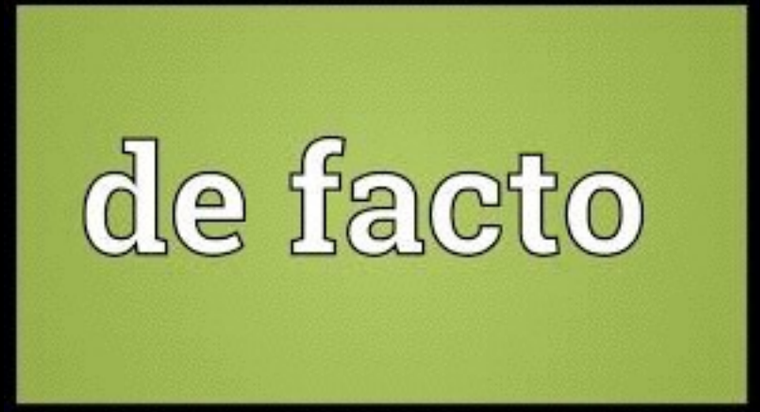
- 로이 필딩(Roy Fielding)

  "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"



REST 원칙의 구성 요소

1. **메서드 (method)**

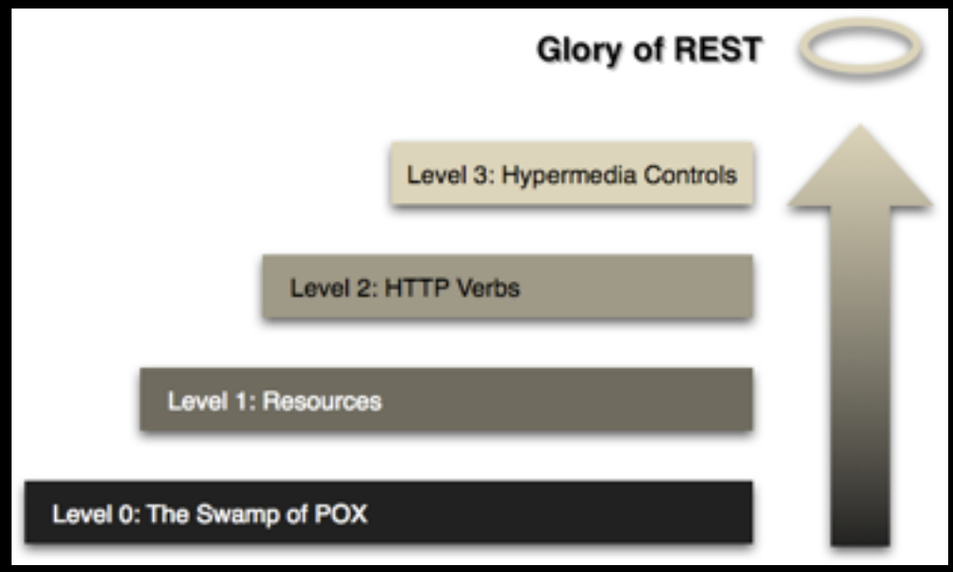
   HEAD, GET, POST, PUT/PATCH, DELETE

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.

```
DELETE /articles/1
GET /deleteArticles?id=1 (X)
```

2. HTTP 메서드 오버라이드

- 일부 브라우저, 네트워크 프록시는 GET, POST만 쓸 수 있다. 해서 PUT(PATCH), DELETE 요청할 때는 서버에 힌트를 제공하는 방법을 제공해야 한다.

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

or

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

3. 리소스는 명사로 표현한다.

- 리소스는 서버에 저장된 데이터(모델)이다.
  URL 엔드포인트는 컬렉션과 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

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

- 필드 이름에도 일관된 규칙 적용

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

5. 관계를 노출할 때는 리소스 중첩

```
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.
- 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": "Hello World"}
```

요점 정리

나는 이제 OOO을 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

더 보면 좋은 자료

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


Stay hungry. Stay foolish.
Steve Jobs
quotefancy