



RESTful API 디자인

막 쓰지 말고 제대로 써보자!

병장 조성수

목차

1. REST API

1-1 요약하면!

2. REST API 특징

2-1 Client-Server

2-2 Stateless Server

2-3 Cache

2-4 uniform Interface

2-5 Layered System

2-6 Code on Demand

3. REST API 디자인

3-1 REST API 디자인 규칙

4. 실제로 써보니

REST API :
요약하자면!

1 REST 란?

REpresentational

Sate

Tansfer

2000년 Roy Fielding의 박사 학위 논문에서 처음 제안

대규모 네트워크 시스템을 위한 아키텍처

REST API :
요약하자면!

1 REST 란?

HTTP URI

HTTP METHOD

REST API :
요약하자면!

1 REST 란?

HTTP URI  HTTP METHOD

REST API :
요약하자면!

1 REST 란?

HTTP URI  HTTP METHOD

- 기존의 웹서비스

`http://hostname/agent.jsp?id=1`

REST API :
요약하자면!

1 REST 란?

HTTP URI  HTTP METHOD

- 기존의 웹서비스

`http://hostname/agent.jsp?id=1`

REST API :
요약하자면!

1 REST 란?

HTTP URI  HTTP METHOD

- 기존의 웹서비스

`http://hostname/agent.jsp?id=1`

- REST 개념 적용

`http://hostname/agent/1 - GET`

REST API :
요약하자면!

1 REST 란?

HTTP URI  HTTP METHOD

- 기존의 웹서비스

`http://hostname/agent.jsp?id=1`

- REST 개념 적용

`http://hostname/agent/1 - GET`

2-1 Client-Server



Client

2-1 Client-Server

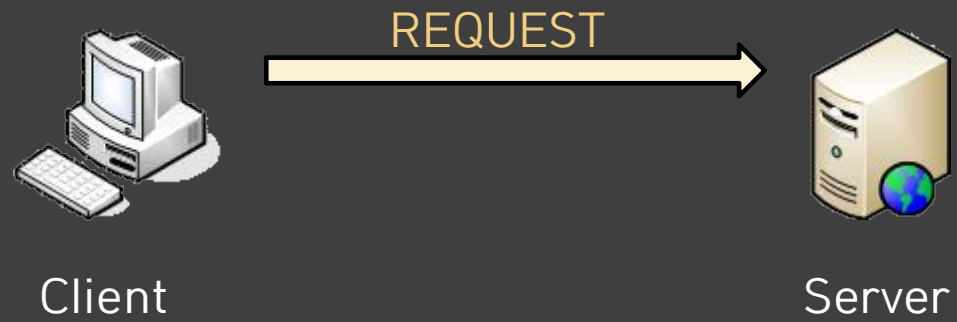


Client



Server

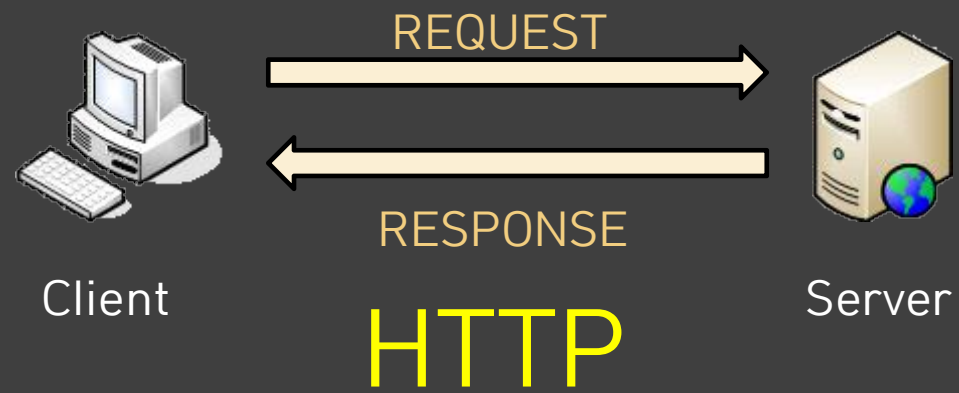
2-1 Client-Server



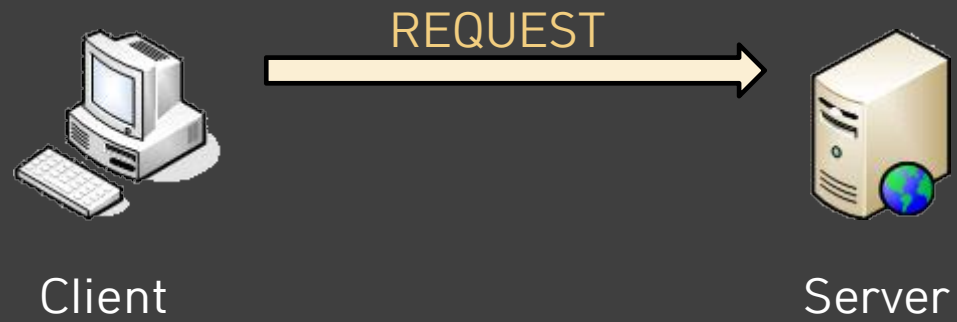
2-1 Client-Server



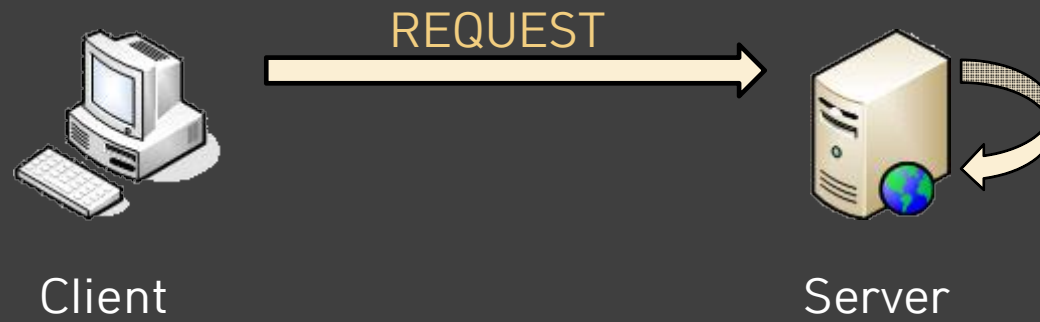
2-1 Client-Server



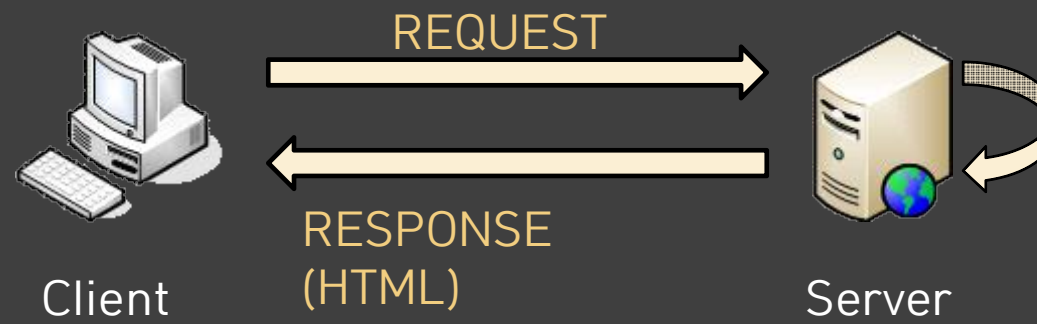
2-1 Client-Server



2-1 Client-Server



2-1 Client-Server



2-1 Client-Server



Client

유저 인터페이스



Server

데이터 스토리지

2-1 Client-Server



2-1 Client-Server



2-1 Client-Server



2-2 Stateless Server

* 서버가 어플리케이션의 상태를 관리하지 않음

NO SESSION

2-3 Cache

- * Client에서 캐시할 수 있도록 함축적/명시적이든 캐시가 가능한 응답은 캐시할 수 있게 제공
- * 물론 서버에서도 가능하면 캐시를 해야함
- * 네트워크 효율 향상

2-4 Uniform Interface

Content-Type : Application/JSON
GET http://domain/resource/1

- * 통일된 URI 접근과 제한된 메소드 인터페이스로 통신
- * 인터페이스의 유연성에 제약을 가함으로써 전체적인 아키텍처가 간결해짐

2-4-1 Identification of Resource

`http://domain/resource/1`

- * REST에서 정보는 Resource고,
resource는 반드시 유일한 URI를 가져야함
- * 웹 기반 REST에서는 resource 접근에 URL 사용

2-4-2 Manipulation of Resources through representations

Content-Type : Application/JSON

GET http://domain/resource/1

- * Resources 와 Presentations 는 구분되어야함
- * Resource는 다양한 형태(XML,JSON, HTML, PNG 등)로 표현될 수 있다.

2-4-3 Self-Descriptive Message

Content-Type : Application/JSON
GET http://domain/resource/1

- * 각 메시지는 반드시 작업을 완료하는데 충분한 정보를 가지고 있어야한다.
- * 웹 기반 REST에서는 Method 와 Header를 활용

3. REST API 디자인

3. REST API 디자인



3. REST API 디자인

• • •
/getAllPigs
/foodNeeded
/createNewPig
/getPig
/massPigParty
/getWhitePig
/healthCheck
/locationVerify
/getWeakPig
/killPig
/eatPig
• • •

3. REST API 디자인

돼지의 세계는 매우 큼니다.

3. REST API 디자인

돼지의 세계는 매우 큼니다.

우리는 단순한 것을 좋아합니다.

3. REST API 디자인

리소스당 2개의 기본 URL을 쓰도록 하겠습니다.

/pigs

/pigs/1234

POST

GET

PUT

DELETE

POST
GET
PUT
DELETE

CREATE
READ
UPDATE
DELETE

Resource	POST create	GET read	PUT update	DELETE delete
/pigs	돼지 추가	돼지들 목록보기	새로운 돼지 리스트로 교체	돼지 전체 삭제
/pigs/1234	돼지 추가	Foo 보기	Foo 정보 업데이트 없으면 생성	Foo 삭제

Resource	POST create	GET read	PUT update	DELETE delete
/pigs	돼지 추가	돼지들 목록보기	새로운 돼지 리스트로 교체	돼지 전체 삭제
/pigs/1234	돼지 추가	Foo 보기	Foo 정보 업데이트 없으면 생성	Foo 삭제

Resource	POST create	GET read	PUT update	DELETE delete
/pigs	돼지 추가	돼지들 목록보기	돼지들 대량 업데이트	돼지 전체 삭제
/pigs/1234	에러	Foo 보기	Foo 정보 업데이트 없으면 에러	Foo 삭제

3-5 디자인 규칙 - Naming

동사는 사용하지 않습니다.

-> HTTP Method 를 동사 대용으로 사용합니다.

3-5 디자인 규칙 - Naming

동사는 사용하지 않습니다.

-> HTTP Method 를 동사 대용으로 사용합니다.

명사를 사용해야합니다.

3-5 디자인 규칙 - Naming

동사는 사용하지 않습니다.

-> HTTP Method 를 동사 대용으로 사용합니다.

명사를 사용해야합니다.

/savePigs

3-5 디자인 규칙 - Naming

동사는 사용하지 않습니다.

-> HTTP Method 를 동사 대응으로 사용합니다.

명사를 사용해야합니다.

~~/savePigs~~

/pigs - POST

3-6 디자인 규칙 - 복수? 단수?

3-6 디자인 규칙 - 복수? 단수?

Foursquare

/chickins

GroupOn

/deals

Zappos

/Product

3-6 디자인 규칙 - 복수? 단수?

복수를 쓰는 것이 더 좋다.

/pigs

3-7 디자인 규칙 - 추상적으로? 구체적으로?

3-7 디자인 규칙 - 추상적으로? 구체적으로?

매우 높은 추상화

/things

높은 추상화

/animals

보통 - 구체적

/pigs

너무 구체적

/wildpig

3-7 디자인 규칙 - 추상적으로? 구체적으로?

추상적인 것 보다 구체적인 것이 좋다.

/pigs

3-8 디자인 규칙 - 에러는 어떻게?



3-8 디자인 규칙 - 에러는 어떻게?

Facebook

[HTTP status Code: 200]

```
{"type":"OAuthException", "message":"(#803 Some of the aliases you requested do not exist: foo.bar)"}
```

Twilio

[HTTP status Code: 401]

```
{"status":401,"message":"Authentication", "code":20003, "more_info":"http://www.twilio.com/docs/errors/20003"}
```

SimpleGEO

[HTTP status Code: 401]

```
{"code":401, "message":"Authentication Required"}
```

3-8 디자인 규칙 - 에러는 어떻게?

Code for code

200 - OK

401 - Unauthorized

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Message for People

```
{"message": "사람들이 알아들을 수 있도록 평범한 설명과 어떻게 해결하는지에 대해 기술한다",  
  "more_info": "더 자세한 설명은 URL로 알려준다"}
```


3-9 디자인 규칙 - 검색

전체 검색

```
/search?q=jeju
```

범위 검색

```
/region/jeju/pigs/search?q=ddong
```

검색 결과 형식 지정

```
/search.json?q=jeju
```

3-10 디자인 규칙 - API URL

Facebook

- * graph.facebook.com
- * api.facebook.com
- * developers.facebook.com

Twitter

- * api.twitter.com
- * search.twitter.com
- * stream.twitter.com
- * dev.twitter.com

실제로 써보니

4. 실제로 써보니

실제로 써보니

4. 실제로 써보니

- * 서버 / 클라이언트(웹) 분업이 편해진다.
- * 클라이언트는 서버가 완료될 때 까지 기다릴 필요가 없다.
- * 개발 속도가 빠르다.
- * 새로운 기능이 생겨도 쉽게 API를 만들고 적용할 수 있다.
- * 테스트가 매우 쉽다.

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

- 학교

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

- 학교
 - └ 1학년
 - └ 2학년
 - └ 3학년

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

- 학교
 - └ 1학년
 - └ 기초생활 수급자
 - └ 불량학생
 - └ 또 다른 그룹
 - └ 2학년
 - └ 3학년

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

예시) 기능 : 사용자 그룹을 구성하라.

- 학교
 - └ 1학년
 - └ 기초생활 수급자
 - └ 불량학생
 - └ 또 다른 그룹
 - └ 2학년
 - └ 3학년

요구 사항 :

- 각 그룹마다 학생들(item)이 n개가 추가된다.
- 그룹은 트리 형태로 표시되며 단계별로 내려간다.

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

- 그룹 경로 구성을 먼저 하자.
- 학교
 - └ 1학년
 - └ 기초생활 수급자
 - └ 불량학생
 - └ 또 다른 그룹
 - └ 2학년
 - └ 3학년

실제로 써보니

4. 실제로 써보니 여러가지 난감한 상황이..

- 그룹 경로 구성을 먼저 하자.
- 학교
 - └ 1학년
 - └ 기초생활 수급자
 - └ 불량학생
 - └ 또 다른 그룹
 - └ 2학년
 - └ 3학년
- 그룹 구성
 - /
 - /학교
 - /학교/1학년
 - /학교/1학년/불량학생
 - /학교/1학년/또 다른 그룹

실제로 써보니

4. 그래서 나온 API

Resource	POST create	GET read	PUT update	DELETE delete
/group/	새로운 그룹 추가	최상위 그룹 리스트 조회	없음	없음
/group/{path}	없음	{path}에 해당하는 그룹 정보 조회	{path}에 해당하는 그룹 정보 업데이트	{path}에 해당하는 그룹 삭제 (하위 그룹 포함)

실제로 써보니

4. 그래서 나온 API

Resource	POST create	GET read	PUT update	DELETE delete
/group/	새로운 그룹 추가	최상위 그룹 리스트 조회	없음	없음
/group/{path}	없음	{path}에 해당하는 그룹 정보 조회	{path}에 해당하는 그룹 정보 업데이트	{path}에 해당하는 그룹 삭제 (하위 그룹 포함)

- * 각 그룹에 학생들은 어떻게 추가시키지??
- * 하위 그룹들은 어떻게 가져오지?

실제로 써보니



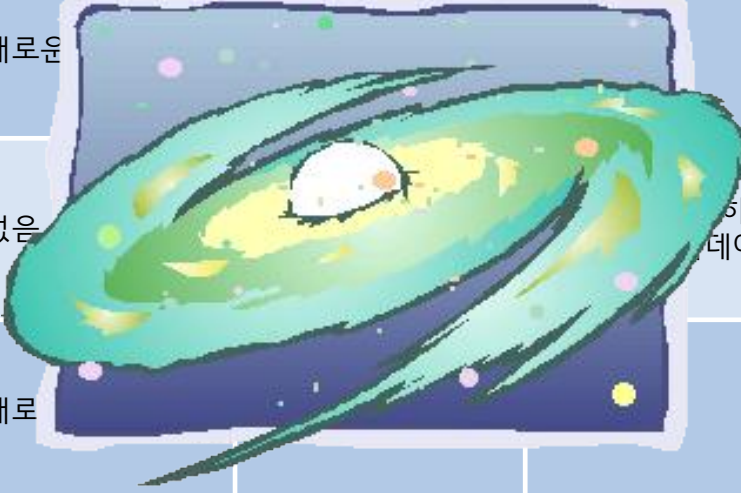
실제로 써 보니

4. 요구사항 반영한 API

Resource	POST create	GET read	PUT update	DELETE delete
/group/	새로운 그룹 추가	최상위 그룹 리스트 조회	없음	없음
/group/{path}	없음	{path}에 해당하는 그룹 정보 조회	{path}에 해당하는 그룹 정보 업데이트	{path}에 해당하는 그룹 삭제 (하위 그룹 포함)
/group/{path}/item	새로운 학생 추가	없음	없음	학생 삭제 (학생 정보는 JSON으로 받음)
/group/{path}/childs	없음	자식 그룹 리스트 조회	없음	없음

실제로 써보니

4. 요구사항 반영한 API

Resource	POST create	GET read	PUT update	DELETE delete
/group/	새로운			없음
/group/{path}	없음			{path}에 해당하는 그룹 삭제 (하위 그룹 포함)
/group/{path}/item	새로			학생 삭제 (학생 정보는 JSON으로 받음)
/group/{path}/childs	없음	자식 그룹 리스트 조회	없음	없음

실제로 써보니

4. 어렵다.

* REST API 원리와 구성은 쉽지만 디자인 원칙에 맞춰서 디자인 하려니 힘든 점이 있었다.

실제로 써보니

4. 어렵다.

- * REST API 원리와 구성은 쉽지만 디자인 원칙에 맞춰서 디자인 하려니 힘든 점이 있었다.
- * 그래도 디자인 원칙은 지키면서 개발자하자.

실제로 써보니

4. 어렵다.

- * REST API 원리와 구성은 쉽지만 디자인 원칙에 맞춰서 디자인 하려니 힘든 점이 있었다.
- * 그래도 디자인 원칙은 지키면서 개발자하자.
- * 왜? 우린 단순 코더가 아닌 개발자니까.

그냥 막 짜도 기능은 동작합니다.

하지만 막 짜면 기능도 막 동작합니다.

코드도 개발도 디자인이 필요합니다..

우리 코드 푸르게 푸르게.

