



RESTful API 설계

REST 이론부터 문서화까지

이동훈 (다행이 아직 20대, 백수무직, leewin12@gmail.com)

목차

1 REST API

1-1 뭐죠?

4 API 문서화

4-1 Swagger

2 API 설계하기

2-1 기본원칙

2-2 가이드라인

3 API 구현하기

3-1 Spring MVC

3-2 JAX-RS 구현

Chapter 1

REST API:

뭐죠?

REST API :
뭐죠?

1 REST가 뭐죠?

REpresentational
Sate
Transfer

by Dr. Roy Fielding에 의해 ‘학위논문’에서 제시
~~이런걸 학위논문... 괜찮아요 우린 천재아닌걸요 데헛~~

REST API :
뭐죠?

1 REST가 뭐죠?

- 아래 6가지를 충족하는 아키텍처 스타일
 - Client-Server
 - Stateless
 - Cache
 - Uniform Interface
 - Layered System
 - Code-On-Demand

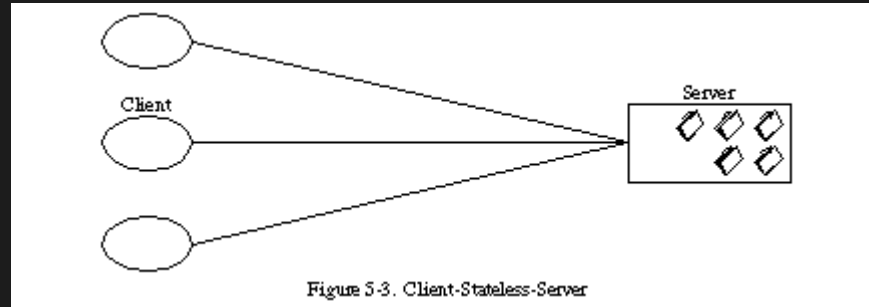
이후 내용들은 주로 아래의 학위 논문에서 끌어왔습니다

Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures, 2000

<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

REST API :
뭐죠?

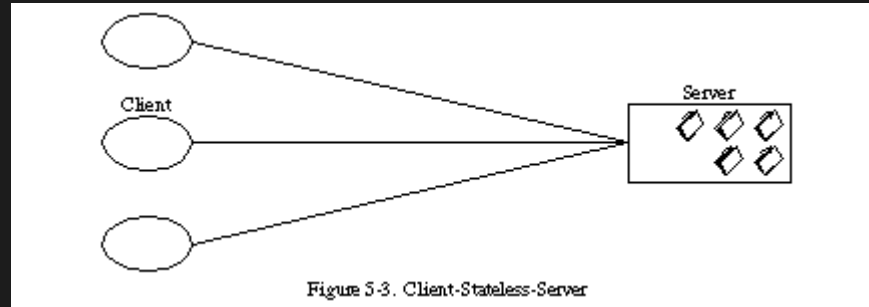
1-1 Client-Server



- 관심사의 분리 (Separation of concerns) 원칙 하의 CS 구조
- 데이터 저장은 Server에서, UI는 Client에서
- 다양한 platform에 대한 이식성 향상
- 서버 컴포넌트의 단순화를 통한 확장성 향상

REST API :
뭐죠?

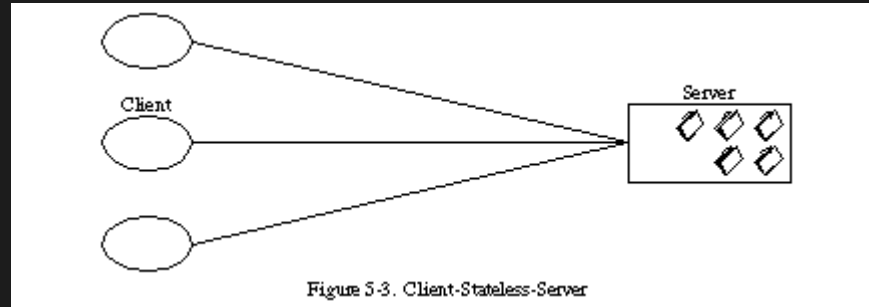
1-2 Stateless



- 각 client 요청은
서버에서 요청을 이해하는데
필요한 모든 정보를 담고 있어야 합니다.
- 사실상 REST 설계의 꽃

REST API :
뭐죠?

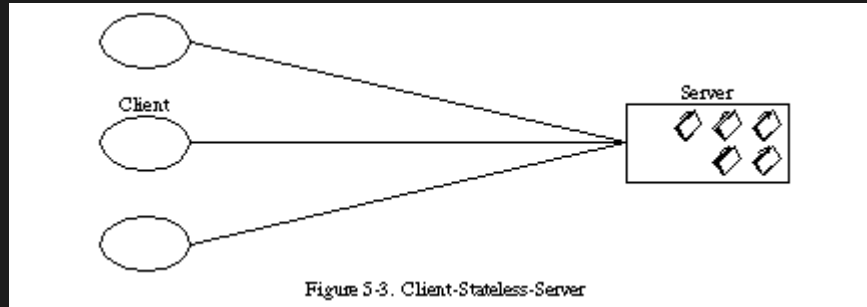
1-2 Stateless



- 따라서 Session state 정보는 완전히 Client에서 관리
- 하지만 현실인 이유로 보안/인증에 대해서는
Token 사용을 인정하는 분위기

REST API :
뭐죠?

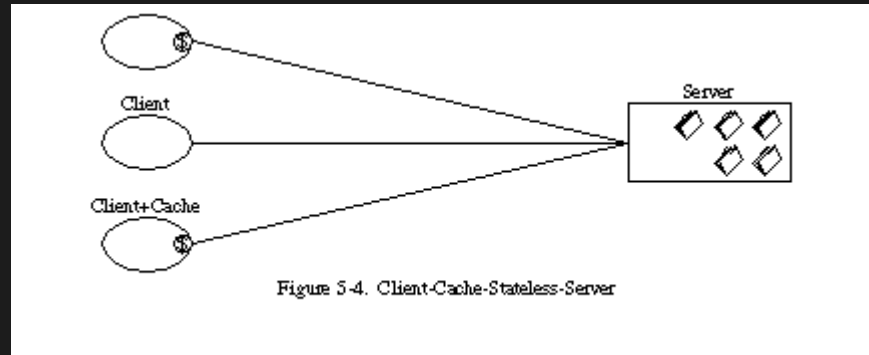
1-2 Stateless



- Session 정보를 공유할 필요가 없으므로,
대단히 효율적인 부하분산과 확장이 가능합니다.
- 정보의 흐름에 대한 이해도 높은 가시성
- 복잡성을 낮춰서 안정성을 높일 수 있음

REST API :
뭐죠?

1-3 Cache



- client에서 캐시할 수 있도록
함축적/명시적이든
캐시가 가능한 응답은 캐시할 수 있게 제공
- 물론 server에서도 가능하면 캐시를 해야함
- 네트워크 효율성 향상

REST API :
뭐죠?

1-4 Uniform interface

Content-type: application/json

GET http://domain/resource/1

- 통일된 URI 접근과 제한된 메소드 인터페이스로 통신
- 주요 4개 제한 조건은 아래와 같음
 - identification of resource
 - manipulation of resources through representations
 - self-descriptive message
 - hypermedia as the engine of application state

REST API :
뭐죠?

1-4-1 Identification of resource

http://domain/resource/1

- REST에서 정보는 resource고,
resource는 반드시 유일한 구분자(URI)를 가져야함
- 웹 기반 REST에서는 resource 접근에 URL 사용

REST API :
뭐죠?

1-4-2 manipulation of resources through representations

Content-type: application/json
http://domain/resource/1

- Resources와 Presentations은 구분되어야함
- Resource는 다양한 형태(XML, JSON, HTML, PNG..)로 표현될 수 있으며, 이를 통해서 조종되어야 합니다.

REST API :
뭐죠?

1-4-3 Self-descriptive Message

Content-type: application/json

GET http://domain/resource/1

- 각 메세지들은 반드시 Task를 완료하는데 충분한 정보를 가지고 있어야 합니다.
- 웹 기반 REST에서는 Method, Header 활용

REST API :
뭐죠?

1-4-4 HATEOAS

REST APIs must be hypertext-driven

Ray T. Fielding, 2008

<http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>

REST API :
뭐죠?

1-4-4 HATEOAS

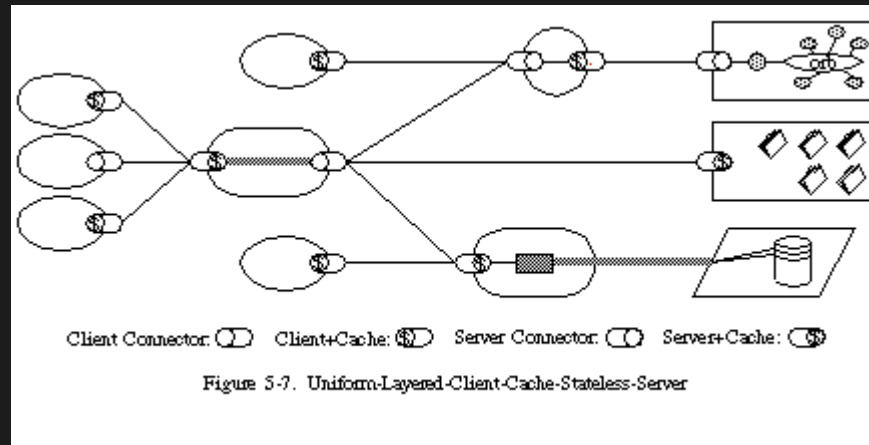
- Hypermedia As The Engine Of Application State
- client는 사전지식 없이 server로부터 동적으로 제공되는 하이퍼미디어를 통해서 상호작용해야함
- 사전 정의된 interface로 통신하는 SOA와 구분
- API구현에 있어서 아직까지 Best practice는 없어보임

예제)

웹브라우저로 조작하는 웹페이지

REST API :
뭐죠?

1-5 Layered System



- Client는 직접 최종서버에 붙었는지 아니면 intermediary 서버에 붙었는지 등을 알 수 없음
- intermediary 서버 등을 통해서 로드밸런싱 / 공유캐시 등을 통해 확장성과 보안을 향상 가능

REST API :
뭐죠?

1-6 Code-On-Demand

- 서버로부터 스크립트를 받아서
client에서 실행하는 걸 말합니다.
- API에서는 많은 곳에서 이야기 하듯 Optional로 봄

예제)

웹페이지의 자바스크립트

Chapter 2

API 설계: 기본원칙

API 설계: 기본원칙

1 전제

- HTTP protocol 기반이 de facto
- URL을 통해서 URI를 표현
- **따라서, URL을 통해서 API를 설계함**

이후 내용들은 주로 아래의 문서에서 끌어왔습니다

Web API Design: Crafting Interfaces that Developers Love

https://blog.apigee.com/detail/announcement_new_ebook_on_web_api_design

API 설계: 기본원칙

2 REST API 설계 기본원칙

- 명사를 쓰고 동사를 피하자
- Resource 당 2개의 기본 URL을 사용하자
- 직관적이고 단순하자

Keep simple things simple

API 설계: 기본원칙

2-1 명사를 쓰고 동사를 피하자

- 실용적인 'RESTful' 한 API 설계에 핵심
- HTTP Method인 POST/GET/PUT/DELETE를 동사 대신에 사용

/getAllLeashedDogs

/verifyVeterinarianLocation

/feedNeedingFood

/createRecurringMedication

2-2 Resource 당 2개의 기본 URL 사용

- Resource에 대해서
 - Collection = /dogs
 - Element in Collection = /dogs/123
- /dog vs /dogs ?
 - 1차 기본 URL은 collection이므로,
/dogs가 되는 것이 더 자연스러움

API 설계: 기본원칙

2-3 종합

| Resource | POST = Create | GET = Read | PUT = Update | DELETE = Delete |
|----------|------------------|---------------|---------------------|--------------------|
| /dogs | 개 추가 | 개 목록보기 | 개 대량 갱신 | 개 전체 삭제 |
| /dogs/12 | 에러 | 개 12 보기 | 개 12 수정 (없으면 에러) | 개 12 삭제 |

- Resource 당 2개 기본 URL
- 명사를 쓰고 동시를 피하기
- POST/GET/PUT/DELETE를 동사 대신에 사용
- 직관적이고 단순하자

Chapter 2

API 설계: 가이드라인

1 URL 표기법 - SnakeCase

/notions/snakes-vs-camel

- '-', '_', '_'를 사용합시다
 - 구글에서 그렇게 권해요
 - IIS는 대소문자 구분 안해요
 - 사람 실수를 방지하기 좋아요

[참고] w3 URL 표준에 따르면

- domain 부분은 대소문자 구분안함
- path 부분은 반드시 대소문자 구분해야함
- IIS를 죽임시다 IIS는 나의 원수

API 설계:
가이드라인



2 JSON 표기법 - CamelCase

JavaScript Object Notation

- JS에선 변수에 대해서는 CamelCase 적용
- 불만은 Douglas Crockford 아저씨께
- Java로 개발하는데 SnakeCase로 하고 싶으면
 - Jackson / Jaxb Annotation 도배질 필요

API 설계: 가이드라인

3 에러처리

- 많은 개발자들이 귀찮아서 넘기는 부분
- 하지만 실제 API를 운용 시, 꼭 신경써야함

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": "Authenticate", "code": 20003, "more  
info": "http://www.twilio.com/docs/errors/20003"}
```

SimpleGeo

HTTP Status Code: 401

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

API 설계: 가이드라인

3-1 Twilio

Twilio

HTTP Status Code: 401

```
{"status" : "401", "message": "Authenticate", "code": 20003, "more  
info": "http://www.twilio.com/docs/errors/20003"}
```

- Http 401 Unauthorized Error
- status: Http Status Code
- message: 상세 에러 메시지
- moreinfo: 에러 참고자료 주소

API 설계: 가이드라인

3-2 HttpStatusCode

- 웬만하면 Http Status Code를 활용합시다
- 다 쓰는 건 교조적, 약 10개 미만으로 선정 사용
- 모두 다 잘됨: 200 OK
- Client에러: 400 Bad Request
- Server에러: 500 Internal Server Error

API 설계: 가이드라인

3-3 HttpStatusCode extend

- 모두 다 잘됨: 200 OK
- Client에러: 400 Bad Request
- Server에러: 500 Internal Server Error

좀 더 써본다면...

- 201 Created
- 304 Not Modified
- 404 Not Found
- 401 Unauthorized
- 403 Forbidden

API 설계: 가이드라인

3-4 에러처리 예제

HTTP Status Code: 400

```
{  
  "developerMessage":  
    "상세하게 개발자의 문제해결을 위한 정보",  
  "userMessage":  
    "사용자용 메시지, 생략가능",  
  "errorCode": 1234,  
  "more info": "http://dev.teach.com/errors/1234"  
}
```


API 설계: 가이드라인

4-1 Versioning

| | |
|----------------|---------------------------------------|
| Twilio | /2010-04-01/Accounts/ |
| salesforce.com | /services/data/v20.0/subjects/Account |
| Facebook | ?v=1.0 |

- Twilio
 - 유럽포맷의 배포 timestamp를 따름
- Salesforce.com
 - 점 사용 표기법으로서 잦은 버전업 암시
- Facebook
 - 강제적으로 버전을 올리려는 의도가 보임

4-2 Versioning 예제

/v1/dogs

- URL의 prefix로 사용
- 번호는 반드시 1,2,3... 정수로만 증가
- 쉽고 간단하고, Spring MVC에서 처리도 쉬움
 - @RequestMapping을 class에 달아 처리

API 설계:
가이드라인

5 페이징과 부분응답

/v1/dogs?

limit=25&offset=50

&fields=id,name,picture

- Param으로 처리
 - 건너뛰기는 offset
 - 갯수제한은 limit
 - Comma delimited list를 사용
 - 기본 값은 정하자

API 설계: 가이드라인

6-1 DateTime

- 사실 JSON에만 발생하는 문제
 - Douglas 아저씨가 표준에 정하지 않았음
- 언어별/라이브러리 별 제각각

API 설계: 가이드라인

6-2 DateTime

언어별 **JSON 라이브러리**의 Date 기본표현

| | |
|-------------|------------------------------|
| ASP.NET | 2013-05-29T15:22:30.9000000Z |
| Json.NET | 2013-05-29T15:22:30Z |
| Jackson | 1369808550900 |
| Gson | May 29, 2013 3:22:30 PM |
| JavaScript | 2013-05-29T15:22:30.900Z |
| Python | 2013-05-29T15:22:30.900000 |
| Objective C | 2013-05-29T15:22:30+09:00 |

… 국제표준 ISO 8601을 쓰면 되는데 무슨 걱정?

API 설계: 가이드라인

6-3 DateTime

언어별 Date 라이브러리의 ISO 8601 기본표현

| | |
|-------------|-------------------------------|
| C# | 2013-05-29T15:22:30.9000000Z |
| JodaTime | 2013-05-29T15:22:30.900+09:00 |
| JavaScript | 2013-05-29T15:22:30.900Z |
| Python | 2013-05-29T15:22:30.900000 |
| Objective C | 2013-05-29T15:22:30+09:00 |

...음.....

API 설계: 가이드라인

6-3 DateTime

언어별 **Date 라이브러리**의 ISO 8601 기본표현

| | |
|-------------|-------------------------------|
| C# | 2013-05-29T15:22:30.9000000Z |
| JodaTime | 2013-05-29T15:22:30.900+09:00 |
| JavaScript | 2013-05-29T15:22:30.900Z |
| Python | 2013-05-29T15:22:30.900000 |
| Objective C | 2013-05-29T15:22:30+09:00 |

...음.....



API 설계:
가이드라인

6-4 DateTime 결론

UnixTime UTC



- 구세주의 등장
- 1970-01-01T00:00:00Z 이후 milisec
- 너무 단순해서 이견이 나올 수 없음
- library/언어에서 똑같은 결과값을 쉽게 추출
- Jackson은 이게 기본!! 진리의 Jackson

API 설계: 가이드라인

6 Collection은 웬만하면 nested로

```
{  
    items: [  
        {}, {}  
    ]  
}
```

- Collection을 그냥 던지면 나중에 확장하려면 골치가 아픕니다.
- 귀찮으면 Generic을 활용

API 설계:
가이드라인

7 ORM Lazy 친구들

`/dogs/1/withA` (X)

`/dogs/1?withA=true` (O)

- Hibernate를 사용할 때 많이 부딪히는 문제
- `/withA`는 path, URI용입니다
- `/withA`는 명사지만, 사실상 동작 역할
- 향후에 `withA`, `withB` 등이 등장하고,
각각/조합 fetch 기능이 요구될 수 있음

API 설계: 가이드라인

8 Id, url

`/dogs/id/{id}` (X)

`/dogs/url/{url}` (X)

`/dogs?url={url}&[name={url}]` (O)

- {url}은 URI가 아님
- url은 동사는 아니지만, 동사 역할
- 만약 url이 prefix 검색으로 바뀐다면?
- 만약 name이 추가 된다면?

API 설계: 가이드라인

9 Login

POST /login (X)

POST /users/login (△)

POST /session (0)

- login은 명사지만, 동사로 사용되고 있음
- login과정 자체도 결과적으로 상태를 추가
- 결과를 주어로 잡고 주어에 대한 동사를 선택

Chapter 3

API 구현하기: Spring MVC

API 구현하기: Spring MVC

1 Spring MVC

- Spring MVC는 범용 Web framework
- REST를 구현하기 위한 많은 기능을 지원
- @RequestMapping
- @PathVariable
- @RequestParam
- @RequestBody
- @ResponseBody
- messageConverter

API 구현하기:
Spring MVC

```
24 @Controller
25 @RequestMapping(value = "/boards/{boardId}/articles")
26 public class ArticleController {
27
28     @RequestMapping(method = RequestMethod.GET)
29     @ResponseBody
30     public ArticleDefaultList list(@PathVariable Long boardId,
31                                   @RequestParam(defaultValue = "0") Integer offset,
32                                   @RequestParam(defaultValue = "5") Integer limit) {
33         return articleService.find(boardId, offset, limit);
34     }
35
36     @RequestMapping(method = RequestMethod.POST, consumes = {"application/json"})
37     @ResponseBody
38     public ResponseEntity<?> add(@PathVariable Long boardId,
39                                  @RequestBody ArticleCreationRequest request) {
40         if (!StringUtils.hasLength(request.getContent())) {
41             return new ResponseEntity<ResponseMessage>(
42                 new ResponseMessage("제목은 비어있을 수 없습니다."),
43                 HttpStatus.BAD_REQUEST);
44         }
45         Article article = articleService.persist(request.toArticle());
46         return new ResponseEntity<Article>(article, HttpStatus.OK);
47     }
48
49     @RequestMapping(value = "/{articleId}", method = RequestMethod.PUT)
50     @ResponseBody
51     public Article edit(@PathVariable Long articleId,
52                        @RequestBody ArticleCreationRequest request,
53                        Principal principal) {
54         return articleService.merge(request.toArticle(principal.getName()));
55     }
56
57     @RequestMapping(value = "/{articleId}", method = RequestMethod.DELETE)
58     public void delete(@PathVariable Long articleId) {
59         articleService.delete(articleId);
60     }
61 }
```

API 구현하기:
Spring MVC

```
24 @Controller
25 @RequestMapping(value = "/boards/{boardId}/articles")
26 public class ArticleController {
27
28     @RequestMapping(method = RequestMethod.GET)
29     @ResponseBody
30     public ArticleDefaultList list(@PathVariable Long boardId,
31                                   @RequestParam(defaultValue = "0") Integer offset,
32                                   @RequestParam(defaultValue = "5") Integer limit) {
33         return articleService.find(boardId, offset, limit);
34     }
35
36     @RequestMapping(method = RequestMethod.POST, consumes = {"application/json"})
37     @ResponseBody
38     public ResponseEntity<?> add(@PathVariable Long boardId,
39                                  @RequestBody ArticleCreationRequest request) {
40         if (!StringUtils.hasLength(request.getContent())) {
41             return new ResponseEntity<ResponseMessage>(
42                 new ResponseMessage("제목은 비어있을 수 없습니다."),
43                 HttpStatus.BAD_REQUEST);
44         }
45         Article article = articleService.persist(request.toArticle());
46         return new ResponseEntity<Article>(article, HttpStatus.OK);
47     }
48
49     @RequestMapping(value = "/{articleId}", method = RequestMethod.PUT)
50     @ResponseBody
51     public Article edit(@PathVariable Long articleId,
52                        @RequestBody ArticleCreationRequest request,
53                        Principal principal) {
54         return articleService.merge(request.toArticle(principal.getName()));
55     }
56
57     @RequestMapping(value = "/{articleId}", method = RequestMethod.DELETE)
58     public void delete(@PathVariable Long articleId) {
59         articleService.delete(articleId);
60     }
61 }
```


API 구현하기:
Spring MVC

```
24 @Controller
25 @RequestMapping(value = "/boards/{boardId}/articles")
26 public class ArticleController {
27
28     @RequestMapping(method = RequestMethod.GET)
29     @ResponseBody
30     public ArticleDefaultList list(@PathVariable Long boardId,
31                                   @RequestParam(defaultValue = "0") Integer offset,
32                                   @RequestParam(defaultValue = "5") Integer limit) {
33         return articleService.find(boardId, offset, limit);
34     }
35
36     @RequestMapping(method = RequestMethod.POST, consumes = {"application/json"})
37     @ResponseBody
38     public ResponseEntity<?> add(@PathVariable Long boardId,
39                                  @RequestBody ArticleCreationRequest request) {
40         if (!StringUtils.hasLength(request.getContent())) {
41             return new ResponseEntity<ResponseMessage>(
42                 new ResponseMessage("제목은 비어있을 수 없습니다."),
43                 HttpStatus.BAD_REQUEST);
44         }
45         Article article = articleService.persist(request.toArticle());
46         return new ResponseEntity<Article>(article, HttpStatus.OK);
47     }
48
49     @RequestMapping(value =("/{articleId}", method = RequestMethod.PUT)
50     @ResponseBody
51     public Article edit(@PathVariable Long articleId,
52                         @RequestBody ArticleCreationRequest request,
53                         Principal principal) {
54         return articleService.merge(request.toArticle(principal.getName()));
55     }
56
57     @RequestMapping(value =("/{articleId}", method = RequestMethod.DELETE)
58     public void delete(@PathVariable Long articleId) {
59         articleService.delete(articleId);
60     }
```

API 구현하기:
Spring MVC

```
24 @Controller
25 @RequestMapping(value = "/boards/{boardId}/articles")
26 public class ArticleController {
27
28     @RequestMapping(method = RequestMethod.GET)
29     @ResponseBody
30     public ArticleDefaultList list(@PathVariable Long boardId,
31                                   @RequestParam(defaultValue = "0") Integer offset,
32                                   @RequestParam(defaultValue = "5") Integer limit) {
33         return articleService.find(boardId, offset, limit);
34     }
35
36     @RequestMapping(method = RequestMethod.POST, consumes = {"application/json"})
37     @ResponseBody
38     public ResponseEntity<?> add(@PathVariable Long boardId,
39                                  @RequestBody ArticleCreationRequest request) {
40         if (!StringUtils.hasLength(request.getContent())) {
41             return new ResponseEntity<ResponseMessage>(
42                 new ResponseMessage("제목은 비어있을 수 없습니다."),
43                 HttpStatus.BAD_REQUEST);
44         }
45         Article article = articleService.persist(request.toArticle());
46         return new ResponseEntity<Article>(article, HttpStatus.OK);
47     }
48
49     @RequestMapping(value =("/{articleId}", method = RequestMethod.PUT)
50     @ResponseBody
51     public Article edit(@PathVariable Long articleId,
52                         @RequestBody ArticleCreationRequest request,
53                         Principal principal) {
54         return articleService.merge(request.toArticle(principal.getName()));
55     }
56
57     @RequestMapping(value =("/{articleId}", method = RequestMethod.DELETE)
58     public void delete(@PathVariable Long articleId) {
59         articleService.delete(articleId);
60     }
```

Chapter 3

API 구현하기: JAX-RS 구현

API 구현하기: JAX-RS 구현

1 JAX-RS

- RESTful 웹서비스를 위한 Java AP
- Annotation 중심으로 정의된 API
- 따라서 다양한 구현체가 존재함
 - Apache CXF: Apache software foundation 구현체
 - Jersey: Sun 표준 구현체
 - RESTeasy: JBoss 구현체
 - Restlet
 - Apache Wink

API 구현하기:
JAX-RS 구현

1 구현체들의 주요특징

- 유연한 연동
 - Spring framework
 - Netty, Grizzly NIO framework
 - Servlet container
- 자체 Client framework을 제공함
- 쉬움, 진짜임

API 구현하기: JAX-RS 구현

```

25 @Controller
26 @Path(value = "/boards/{boardId}/articles")
27 @Produces(MediaType.APPLICATION_JSON)
28 @Consumes({ MediaType.APPLICATION_JSON })
29 public class HomeController {
30     @GET
31     public ArticleDefaultList list(@PathParam("boardId") Long boardId,
32                                   @QueryParam("offset") @DefaultValue("0") Integer offset,
33                                   @QueryParam("limit") @DefaultValue("5") Integer limit) {
34         return articlesService.find(boardId, offset, limit);
35     }
36     @POST
37     public Response add(@PathParam("boardId") Long boardId,
38                        ArticleCreationRequest request) {
39         if (!StringUtils.hasLength(request.getContent())) {
40             return Response.status(Status.BAD_REQUEST)
41                 .entity(new ResponseMessage("제목은 비어있을 수 없습니다"))
42                 .build();
43         }
44         Article article = articleService.persist(request.toArticle());
45         return Response.ok(article).build();
46     }
47
48     @PUT
49     @PathParam("/{articleId}")
50     public Article edit(@PathParam("articleId") Long articleId,
51                        ArticleCreationRequest request, @Context Principal principal) {
52         return articleService.merge(request.toArticle(principal.getName()));
53     }
54
55     @DELETE
56     @PathParam("/{articleId}")
57     public void delete(@PathParam("articleId") Long articleId) {
58         articleService.delete(articleId);
59     }

```

API 구현하기: JAX-RS 구현

```

25 @Controller
26 @Path(value = "/boards/{boardId}/articles")
27 @Produces(MediaType.APPLICATION_JSON)
28 @Consumes({ MediaType.APPLICATION_JSON })
29 public class HomeController {
30     @GET
31     public ArticleDefaultList list(@PathParam("boardId") Long boardId,
32                                   @QueryParam("offset") @DefaultValue("0") Integer offset,
33                                   @QueryParam("limit") @DefaultValue("5") Integer limit) {
34         return articleService.find(boardId, offset, limit);
35     }
36     @POST
37     public Response add(@PathParam("boardId") Long boardId,
38                        ArticleCreationRequest request) {
39         if (!StringUtils.hasLength(request.getContent())) {
40             return Response.status(Status.BAD_REQUEST)
41                 .entity(new ResponseMessage("제목은 비어있을 수 없습니다"))
42                 .build();
43         }
44         Article article = articleService.persist(request.toArticle());
45         return Response.ok(article).build();
46     }
47
48     @PUT
49     @PathParam("/{articleId}")
50     public Article edit(@PathParam("articleId") Long articleId,
51                       ArticleCreationRequest request, @Context Principal principal) {
52         return articleService.merge(request.toArticle(principal.getName()));
53     }
54
55     @DELETE
56     @PathParam("/{articleId}")
57     public void delete(@PathParam("articleId") Long articleId) {
58         articleService.delete(articleId);
59     }

```


API 구현하기: JAX-RS 구현

```

25 @Controller
26 @Path(value = "/boards/{boardId}/articles")
27 @Produces(MediaType.APPLICATION_JSON)
28 @Consumes({ MediaType.APPLICATION_JSON })
29 public class HomeController {
30     @GET
31     public ArticleDefaultList list(@PathParam("boardId") Long boardId,
32                                   @QueryParam("offset") @DefaultValue("0") Integer offset,
33                                   @QueryParam("limit") @DefaultValue("5") Integer limit) {
34         return articleService.find(boardId, offset, limit);
35     }
36     @POST
37     public Response add(@PathParam("boardId") Long boardId,
38                        ArticleCreationRequest request) {
39         if (!StringUtils.hasLength(request.getContent())) {
40             return Response.status(Status.BAD_REQUEST)
41                 .entity(new ResponseMessage("제목은 비어있을 수 없습니다"))
42                 .build();
43         }
44         Article article = articleService.persist(request.toArticle());
45         return Response.ok(article).build();
46     }
47
48     @PUT
49     @PathParam("/{articleId}")
50     public Article edit(@PathParam("articleId") Long articleId,
51                       ArticleCreationRequest request, @Context Principal principal) {
52         return articleService.merge(request.toArticle(principal.getName()));
53     }
54
55     @DELETE
56     @PathParam("/{articleId}")
57     public void delete(@PathParam("articleId") Long articleId) {
58         articleService.delete(articleId);
59     }

```

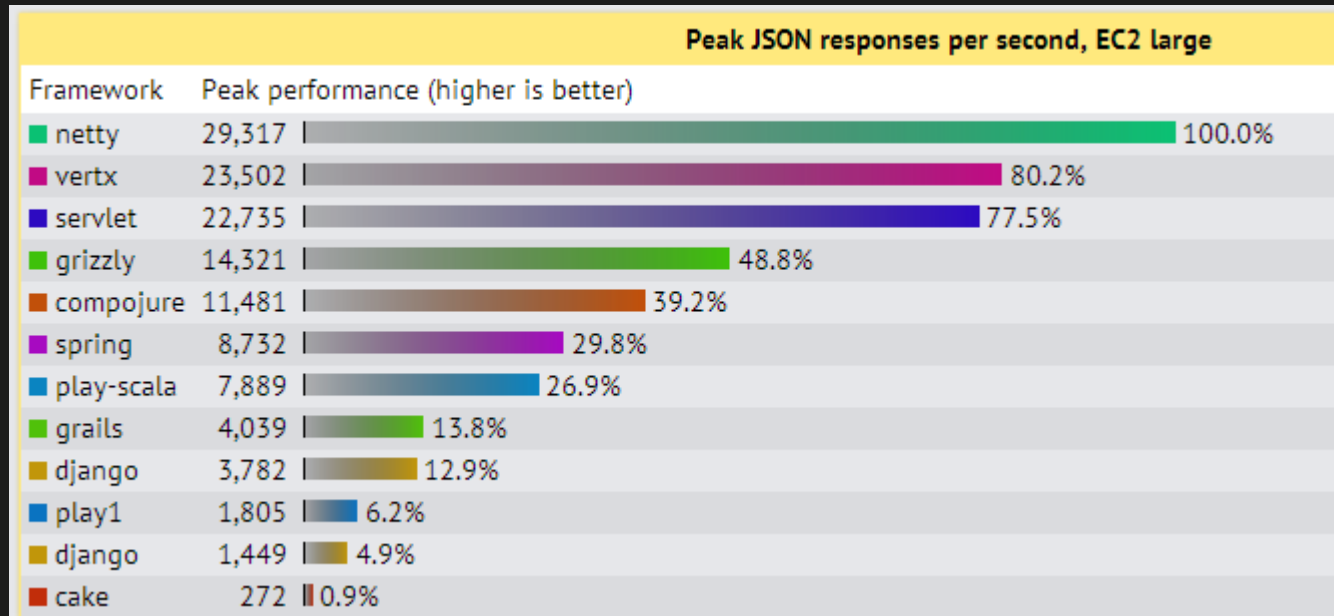

API 구현하기:
JAX-RS 구현

2 Why JAX-RS annotation

- 단순함
 - @Provider로 대부분의 설정/추가로직 제어
- Servlet container 없이도 동작가능
 - Netty, Grizzly 등등
- 편리한 Request/ResponseBuilder
- REST 관련 기술의 빠른 도입
 - Gzip, Cache, OAuth, WebSocket 등
- 많은 구현체, 많은 도구들
 - Swagger 문서자동화툴

API 구현하기: JAX-RS 구현

[참고용] WebFramework Benchmark



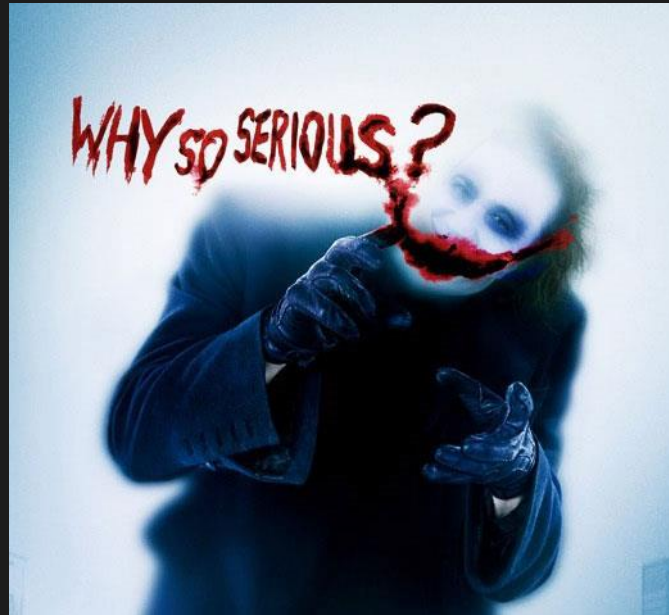
- DB 때고 JSON serialization response 처리만
- 이의는 Github pull request로 받겠습니다

출처: <http://www.techempower.com/benchmarks/#section=data-r5&l=8ti&f=hkmv0e-cquxoo-1>

Chapter 4

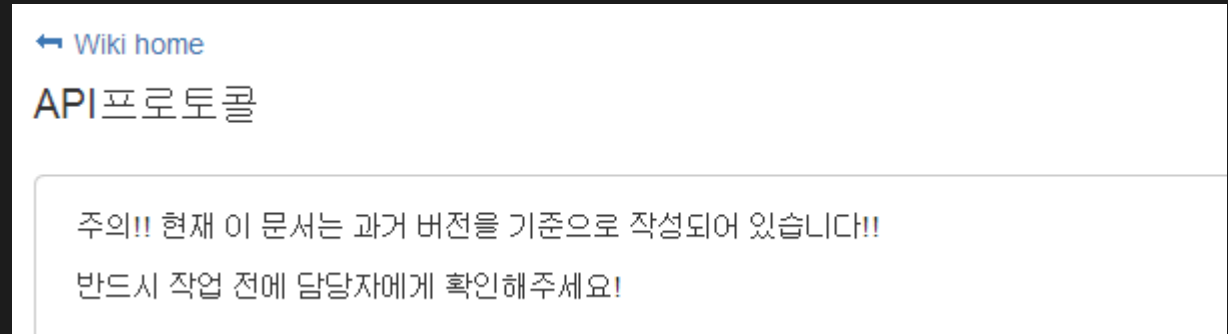
API 문서화: Swagger

API 문서화: Swagger



Why so need it?

API 문서화: Swagger






나만 그런거야?

그런거야?

왜 만날 문서는 업데이트 안하는겨...

API 문서화: Swagger


swagger



Explore

/user

Show/Hide | List Operations | Expand Operations | Raw

POST /user.json/createWithArray

Creates list of users with given input array

Parameters

| Parameter | Value | Description | Data Type |
|-------------|------------|---------------------|---|
| Array[User] | (required) | List of user object | <div>Model Model Schema</div> <div> Array[User] class User(id: long, lastName: string, phone: string, username: string, email: string, userStatus: int = ['1-registered' or '2-active' or '3-closed'] {User Status}, firstName: string, password: string) </div> |

Parameter content type:

Try it out!

POST /user.json

Create user

POST /user.json/createWithList

Creates list of users with given list input

PUT /user.json/{username}

Updated user

DELETE /user.json/{username}

Delete user

GET /user.json/{username}

Get user by user name

GET /user.json/login

Logs user into the system

GET /user.json/logout

Logs out current logged in user session

/pet

Show/Hide | List Operations | Expand Operations | Raw

API 문서화:
Swagger



0치고 제 돈 가져요

API 문서화:
Swagger

근데 사실 우린 저런거 필요 없어요

1) Maven 연동

2) Wiki / markdown 생성

API 문서화: Swagger

About the template file

Don't worry about the template file, the plugin has embed 3 templates in:

1. wiki.mustache for wiki markup output.
2. html.mustache for html output.
3. markdown.mustache for markdown output.

You can directly use them in **outputTemplate**, no extra path is needed, just the filename will be the path. e.g.:

```
<outputTemplate>markdown.mustache</outputTemplate>
```

If you dissatisfied with the included mustache template, you can write your own, just follow the following json-schema of the hash your mustache template file will consume. It looks big but actually simple:

```
{
  "type": "object",
  "properties": {
    "basePath": {
      "type": "string"
    },
    "apiVersion": {
      "type": "string"
    },
    "apiDocuments": {
```



```

31 @Api(value = "/boards/{boardId}/articles", description = "article관련 api들")
32 @Controller
33 @Path(value = "/boards/{boardId}/articles")
34 @Produces(MediaType.APPLICATION_JSON)
35 @Consumes({ MediaType.APPLICATION_JSON })
36 public class HomeController {
37
38     @GET
39     @ApiOperation(value = "목록보기",
40         responseClass = "org.greg.resteasy.controller.ArticleDefaultList")
41     public ArticleDefaultList list(@PathParam("boardId") Long boardId,
42         @QueryParam("offset") @DefaultValue("0") Integer offset,
43         @QueryParam("limit") @DefaultValue("5") Integer limit) {
44         return articleService.find(boardId, offset, limit);
45     }
46
47     @POST
48     @ApiOperation(value = "신규추가")
49     @ApiResponse(value = "추가된 객체",
50         errors = { @ApiError(code = 400, reason = "제목이 비어 있는 경우") })
51     public Response add(
52         @ApiParam("추가하고자 하는 게시판 Id") @PathParam("boardId") Long boardId,
53         @ApiParam("신규정보") ArticleCreationRequest request) {
54         if (!StringUtils.hasLength(request.getContent())) {
55             return Response.status(Status.BAD_REQUEST)
56                 .entity(new ResponseMessage("제목은 비어있을 수 없습니다"))
57                 .build();
58         }
59         Article article = articleService.persist(request.toArticle());
60         return Response.ok(article).build();
61     }

```

API 문서화:
Swagger

그럼 어떻게 나와요??

Resources

1. /boards/{boardId}/articles

Overview

article관련 api들

1.1 /boards/{boardId}/articles

1.1.1 add

POST /boards/{boardId}/articles

신규추가

URL

http://host:port/foo/bar/boards/{boardId}/articles

Parameters

- body

| Parameter | Required | Description | Data Type |
|------------------------|----------|-------------|--|
| ArticleCreationRequest | false | 신규정보 | ArticleCreationRequest |

- path

| Parameter | Required | Description | Data Type |
|-----------|----------|-----------------|-----------|
| boardId | true | 추가하고자 하는 게시판 Id | long |

- path

| Parameter | Required | Description | Data Type |
|----------------|----------|-------------|-----------|
| boardId | true | | long |

Response

[ArticleDefaultList](#)

Errors

| Status Code | Reason |
|-------------|--------|
|-------------|--------|

Data Types

ArticleCreationRequest

| type | required | access | description | notes |
|----------------|----------|----------|-------------|-------|
| content | string | optional | - | - |

ArticleDefaultList

| type | required | access | description | notes |
|--------------|-------------------------------|----------|-------------|-------|
| items | Array:Article | optional | - | - |

API 문서화: Swagger

장점

- 개발과정에서 API 버그 찾기가 매우 편함
- 비교적 문서가 항상 최신으로 유지 가능
- 코드에 문서가 포함되어 있으므로 직관적

단점

- 사실 wiki용은 안 이빠서 스샷 안찍음
 - 소소하게 안되는 부분이 있음
- 하지만 대부분 issue 발행되어 개발진행 중
그리고...

API 문서화:
Swagger



사실 Spring MVC은 안되요

API 문서화: Swagger

Swagger-spring-mvc 있잖아요

- Spring MVC를 실제로 띄워서 runtime에 return되는 json을 잡아서 이쁘게 처리해요
- 그래서 mvn은 바이바이 standalone 아듀

언젠간 될 거예요...

- 이슈가 올라와서 열심히 토론 중이에요
- 언젠간 될 거예요
- 개발자가 좋은 아이디어 있으면 달라고 하네요

끝

진지한 궁서체