Remotty Tech Blog

RestFul이란 무엇인가?

이 글에서는 **REST(Representational State Transfer)**에 대해서 알아보겠습니다.

목차

머리말

바로 위에서 REST에 대해서 알아본다고 하였지만, **REST의 정의와 같은 것은 생략할 예정**입니다. 진짜로 이글에서 다룰 것은 실제 **RESTFul한 API를 작성할 때 도움될만한 것들을 공부**합니다. 또한 여러가지 규칙이 있지만 어느 규칙이 진짜이고, 표준화 된것이 없기때문에(이런것으로 알고있습니다.) 실제 많은 사이트들이 약간씩은 다른 형태로 REST API를 운영하고 있습니다. 이 글에서도 **필자가 중요하다고 생각하고 직관적이라고 생각하는 요소들을 선택 하여 소개 또는 설명 할 예정**입니다.

요즘은 예전과 달리 소비를 위한 장비 및 프로그램이 다양합니다. 반대로 말하면 예전에는 어떠한 인터넷 컨텐츠를 소비하기위한 장비나 프로그램이 몇개 없었습니다. 그렇기때문에 서버와 클라이언트가 거의 1:1이였습니다. 그러나 요즘은 역시 서버는 1인데 클라이언트가 굉장히 다양해졌습니다. 안드로이드는 OS 버전도 굉장히 다양하고 단말기마다 굉장히 다른 특성을 갖기도 합니다. 또 IOS도 있고, 컴퓨터 브라우져의 종류도 많아졌죠.

그래서 예전처럼 **하나의 클라이언트를 위한 서버를 구성하는건 비효율적인 일**이 되어버렸습니다. 하나의 서버로 여러대의 클라이언트를 대응하도록 할때 필요한것이 RESTFul API입니다.

URI 설계하기

일단 URI(Uniform Resource Identifier)란 영어 약자를 풀어보면 '균등한 리소스 식별자'정도로 할 수 있습니다. 말그대로 인터넷의 어떠한 리소스를 식별하기 위해서 만들어진 것입니다. 잘 감이 안오는데 중간중간 특성들을 이야기하면서 내용을 보강하겠습니다.

밑에서 설명하는 규칙들은 RFC3986을 기반으로 작성되었습니다.

소문자를 사용하자(최소한 대소문자를 구분한다는 사실을 알고있자)

```
http://www.example.com:80/users/1?q=abc#title
|_____________________|_____|_____|
          1              2     3
URI는위와같이 크게 3부분으로 나뉘어져있습니다.
rfc3986에서는 1번부분(host와 port)을 제외한 2번 3번 부분은 대소문자를 구분하도록 하였습니다.
```

즉

```
a - http://abc.com/haha
b - HTTP://ABC.COM/haha
c - http://abc.com/HAHA
d - http://abc.com/hAhA
```

위 예제에서 a와 b만 같은 리소스이고 나머지 조합은 전부 다른 리소스입니다. 그러므로 대소문자를 섞어서 사용하는건 혼란을 가져올 수 있으므로 지양해야겠습니다. 그래도 URI에 사람이름과 같은 고유명사가 들어갈 수 도있죠.. 이땐 대문자를 쓰고싶은데… 사용해도 되긴합니다만 대소문자가 구분된다는걸 이해하셔야 합니다. 위 설명과 중복되긴 하지만 다시한번 언급하겠습니다.

```
http://www.remotty.com/countries/korea
http://www.remotty.com/countries/Korea
```

위 두개의 URI는 분명히 다른 리소스를 가리키고 있는 것입니다.

다만 국가코드를 포함하여 도메인을 의미있게 하기위해서 이와 같은 형태로 사용하는 것은 굉장히 좋다고 봅니다. `http://cleanHo.me`, `http://adBy.me`, `http://dishBy.me`, `http://googleSear.ch` 이 경우에도 경로(path)부분은 소문자를 사용하는게 좋겠습니다.

하이픈(-, hyphen)을 사용하자

경로(path)에 공백(띄어쓰기)가 들어갈땐 경우에따라 띄어쓰기 대신 `%20`이 쓰일때가 있습니다. 이런건 보기에 별로 좋지 않기때문에 많은 사람들이 밑줄(_, underscore) 또는 하이픈(-, hyphen)을 사용하여 공백을 대체합니다. 여기서 권장하는건 하이픈만 사용하자 입니다. 보통 밑줄은 링크가 걸린부분에 표시되는데 그것과 중복되면 리소스로써의 밑줄은 가려질 수도있기때문에 하이픈을 사용하자.

확장자를 사용하지 말자

기존의 많은 URI들이 확장자를 포함하고 있습니다. 하지만 REST API를 설계할때에는 확장자는 사용하지 않는게 좋겠습니다. 확장자를 사용하지 않으면 리소스가 더 유연해집니다. 어째든 확장자를 사용하지 않는다면 기존에 REST API를 접해보지 않는 사람들은 이런 의문을 갖을 수 있습니다? 결론적으로 Accept header를 사용해야 합니다. 예를들어 내용이 `Hello,World`인 파일이 있습니다. 서버를 기존방식대로 설계한다면 해당 파일은 `http://remotty.com/hello.txt`와 같이 요청하여 응답받을 것입니다. 기존의 방식은 분명하게 파일의 형태가 txt로 고정되어있습니다. csv 형태로도 제공하려면 `http://remotty.com/hello.csv` URI도 준비해야 할것이고, 서버엔 hello.txt와 hello.csv 두개의 파일이 존재 하게 될것입니다. `http://remotty.com/hello.txt`와 `http://remotty.com/hello.csv`는 분명하게 다른 리소스를 식별하

는 URI이지만, 실제론 하나의 리소스를 가르키고 있습니다. 이것은 비 효율적입니다. 리소스가 한개라면 URI도 한개여야합니다.

REST API에서는 `http://remotty.com/hello`에 대한 대응만 해놓고, 해당 요청이 왔을때 `Accept` header를 적절히 파싱(parsing)하여 클라이언트(client)가 요청한대로 응답해주면 됩니다.

REST API로 구현했을땐

```
GET /hello HTTP/1.1
Host: remotty.com
Accept: text/plain
```

또는

```
GET /hello HTTP/1.1
Host: remotty.com
Accept: text/csv
```

`Accept`를 좀더 적극적으로 활용한다면 이렇게도 가능합니다.

```
GET /hello HTTP/1.1
Host: remotty.com
Accept: text/csv,text/html;q=0.5,application/xml;q=0.6,text/plain;q=0.9,application/pdf,*/*;q=0.3
```

`Accept` header은 클라이언트가 자신이 선호하는 media type을 서버에 보내는 것입니다. 서버에 위 예제와 같은 요청이 왔다면 가장먼저 우선순위는 다음과 같습니다.

1. text/csv
2. application/pdf
3. text/plain;q=0.9
4. application/xml;q=0.6
5. text/html;q=0.5
6. */*;q=0.3

q가 생략된것은 자동으로 1로 설정되며, q의 범위는 0부터 1입니다. 서버가 판단하여 csv로 응답을 할 수 있다면 csv로 응답을 하고 csv가 준비가 되지 않았다면 그다음 우선순위인 pdf로 pdf도 준비되지 않았다면 계속 다음으로 넘어가다가 서버가 응답할수있는 media type가 `Accept`에 명시되지 않았다면 http status code 406과 body엔 응답 가능한 media type를 명시하여 응답하여야 합니다. 더 자세한 내용은 `http content negotiation`으로 검색하시어 저도 좀 알려주시기 바랍니다.

위와 같이 `Accept` header를 적절히 잘 사용하면 하나의 URI로 클라이언트의 요청에 대한 응답을 좀더 유연하게 할 수 있습니다.

CRUD는 URI에 사용하면 안된다

과거에 `GET`, `POST`만 사용하였을땐 CRUD를 URI에 표시해주어야 했습니다.

예를들면

```
GET /posts/13/delete HTTP/1.1
POST /posts/write HTTP/1.1
GET /delete?id=55 HTTP/1.1
```

하지만 당신은 이제부터 REST API를 설계할 수 있습니다. 뒷부분에 나오는 HTTP Method의 알맞은 역할을 참조하여 적절한 Method를 사용하여 CRUD는 URI에 사용하지 않기로 합시다.

컬렉션과 도큐먼트

이 글을 보기전 REST 관련된 글을 본적이 있다면 컬렉션(collection)이라고 이야기하는 도큐먼트(document)들의 집합을 들어보았을 겁니다. 도큐먼트는 우리말로 문서로 이해해도 되고, 정보라고 이해해도 무관합니다. 도큐먼트는 엘리먼트(element)라고도 하더라고요. 컬렉션은 정보(문서)들의 집합이 이라고 할 수 있겠네요. 컬렉션과 도큐먼트는 모두 리소스라고 표현할 수 있으며 URI에 나타납니다.

예제를 통해 이해도를 높여보겠습니다.

http://www.remotty.com/sports/soccer
http://www.remotty.com/sports/soccer/players
http://www.remotty.com/sports/soccer/players/13/skills

위 URI를 요청하면 어떤 응답이 오게 될지 대충 예감이 오시나요…?

http://www.remotty.com/sports는 컬렉션입니다. sports컬렉션에 soccer도큐먼트가 있는거고요. 또 soccer도큐먼트에 player이라는 컬렉션이 존재하는겁니다! players 컬렉션에 뭐 등번호가 13번인 선수가 있나봅니다. 여기서 13은 도큐먼트이고요. 아시겠죠…? 조금 더 이야기 해보겠습니다. soccer 도큐먼트와 동일한 수준의 다른 도큐먼트는 뭐가 있을까요…? baseball, marathon 등이 있겠네요. 그 하위의 players의 컬렉션과 동일한 수준의 컬렉션은 뭐가있을까요…? rules, leagues 등이 있겠네요. 이제 뭔지 아시겠죠…?

여기서… 중요한 법칙이랄꺼 까진 없지만 뭔가 있습니다. 바로 컬렉션은 복수로 사용하네요. 어쩌면 당연하지만 직관적인 REST API를 위해선 단수 복수도 정확하게 지켜주면 좋겠습니다. 요즘은 한글이 URI에 많이 들어가는데 players는 선수들로 하면 되려나요…?

HTTP Method의 알맞은 역할

HTTP Method는 여러가지가 있지만 REST API에서는 4개 혹은 5개의 Method만 사용됩니다. POST, GET, PUT, DELETE 이 4가지의 Method를 가지고 CRUD를 할 수 있습니다. 그러나 REST API에서 사용되는 개수는 4개 혹은 5개라고 한 이유는 PATCH를 포함하면 5개가 됩니다.

각 Method마다 올바른 역할이 있습니다. 아래 표를 보면서 이해를 높이겠습니다.

URI	POST	GET	PUT	DELETE
http://www.remotty.com/sports	1	2	-	-
http://www.remotty.com/sports/soccer	-	3	4	5

text table generator에서 만든 text table인데 한글을 사용하면 깨지므로… :(

번호를 이용하여 설명하겠습니다.

1. 현재 리소스 보다 한단계 아래에 리소스를 생성합니다. POST Method를 통해 해당 URI를 요청하면 sports 컬렉션에 알맞은 soccer 또는 baseball과 같은 도큐먼트 리소스를 생성합니다.
2. 현재 리소스를 조회합니다. 보통 컬렉션 리소스를 조회하게되면 하위의 도큐먼트들의 목록과 아주 간단한 정보들을 가져옵니다.
3. 현재 리소스를 조회 합니다. 도큐먼트 리소스를 조회하게되면 해당 도큐먼트에 대한 자세한 정보들을 가져옵니다.
4. 현재 리소스를 수정합니다. soccer에 대한 정보를 수정하게 됩니다.
5. 현재 리소스를 삭제합니다. DELETE Method를 이용하여 현재 URI를 호출하면 sports 컬렉션에서 soccer 도큐먼트가 삭제됩니다.

※추가로 주목해볼 만한 Method는 PATCH입니다. 기존에 REST에 익숙하신분들은 수정(update)을 위한 Method는 PUT가 익숙하실 겁니다. 하지만 앞으로는 PUT대신 PATCH를 자주 써야할꺼 같아요. 자세한 설명은 Edge Rails: PATCH is the new primary HTTP method for updates을 참조해주세요.

반응형 웹에서의 REST

요즘은 정말 다양한 장비들이 존재합니다. 크기 역시 다양합니다. 그래서 요즘 작은 화면의 기기로 포털사이트를 접속해보면 어느 형태로든 m이 붙은걸 볼수 있습니다. 아마 mobile에서 맨앞의 m인것같은데, 화면이 작은 장비에서는 800*600이 넘어가는 사이트를 돌아다니는건 정말 피곤한 일입니다. 그래서 작은 화면용 페이지가 필요한 이유입니다. 네, 여기까지는 좋습니다. 제가 이 부분에서 소개해드리려는 header는 User-Agent입니다. 그리고 결론적으로 말씀드리고 싶은 내용은 http://m.remotty.com/abc 또는 http://www.remotty.com/m/abc처럼 사용하지 말자 입니다.

한가지 예를 들어보겠습니다.

1. 철수가 안드로이드로 뉴스를 읽고있음

2. 재미있는 기사 발견,
3. sns로 공유
4. http://m.remotty.com/fun 공유됨.
5. 영희가 이번에 새로산 신상 15형 MacBook Pro Retina을 가지고 sns 보고있음
6. 철수가 공유한 링크(http://m.remotty.com/fun)클릭.
7. 영희는 작은화면으로 웹서핑을 하고있는게 아닌데도 불구하고
8. 모바일에 최적화된 화면으로 뉴스를 읽게됩니다.

뭐가 문제인지 감이 오시나요? 이미 많은 웹사이트가 `User-Agent` header을 사용중입니다. `User-Agent` header을 적절히 파싱하여 화면이 작은 장비는 모바일에 최적화된 사이트로 이동(redirect)시켜주고 있습니다.

RESTFul한 웹사이트에서는 `User-Agent`를 이용하여 다른 곳으로 리다이렉트 시켜주는게 아니라 URI는 그대로이지만 화면만 장비에따라 알아서 최적화 되도록 설계해야합니다.

다시말해서 http://www.remotty.com/info와 http://m.remotty.com/info는 사실상 같은 정보를 보여주고 있지만 화면의 형태만 다를 뿐입니다. REST하게 설계할땐 리소스가 같다면 URI는 하나여야 합니다.

I18n과 REST

현재 많은 웹사이트들이 다국어를 지원하고있고, http://www.remotty.com/ko/info나 http://en.remotty.com등과 같이 언어마다 다른 URI를 운영중인 곳이 있습니다. 여기서 소개해드릴 것은 Accept-Language header입니다.

또 한번 예를 들어보겠습니다.

1. 한국인 철수가 뉴스를 읽음.
2. 재미있는 기사발견,
3. sns로 공유
4. http://ko.remotty.com/fun 공유됨.
5. remotty는 다국어가 아주 잘 지원되는 사이트임.
6. 미국인 Cathy(캐씨)가 sns를 보고있음
7. 캐씨는 미국인임에도 불구하고,
8. 한국어로된 뉴스기사를 읽게됩니다.

뭐가 문제인지 감이 오시나요? 다국어 지원을 `Accept-Language`에 맡긴다면 URI는 그대로인데, 사용자의 환경에 따라 알맞은 언어로 응답할 수 있습니다. 물론 `Accept-Language`만 가지고 다국어를 하면 조금 어색할 수 있을꺼 같습니다. 사용자가 원하는 언어를 설정하게하여 해당 언어를 세션 또는 쿠키 등에 저장하여 보여줄 언어를 선정할때 우선순위를 약간 조정하여 보여주는게 좋은 방법일꺼 같습니다.

지금 약간 이글의 확장자를 사용하지 말자와 반응형 웹에서의 REST랑 약간 비슷한 느낌인데요, 모두 결론은 URI는 리소스를 식별하기 위해서 사용되었지, 리소스가 어떻게 보여지느냐는 별도의 header을 이용하여 처리하였습니다. 이 점을 생각하면서 다른 문제들도 좀더 REST하게 설계해야겠습니다.

응답 상태 코드

rfc2616을 살펴보면 많은 종류의 상태코드가 존재합니다. 상태코드를 적절히 잘 사용하면 클라이언트에게 많은 정보를 줄 수 있습니다.

성공

- 200 - 클라이언트의 요청을 정상적으로 수행하였을때 사용합니다. 응답 바디(body)엔 요청과 관련된 내용을 넣어줍니다. 그리고 200의 응답 바디에 오류 내용을 전송하는데 사용해서는 안된다고 합니다. 오류가 났을땐 40x 응답코드를 권장합니다.
- 201 - 클라이언트가 어떤 리소스 생성을 요청하였고, 해당 리소스가 성공적으로 생성되었을때 사용합니다.
- 202 - 클라이언트의 요청이 비동기적으로 처리될때 사용합니다. 응답 바디에 처리되기까지의 시간 등의 정보를 넣어주면 좋다고 합니다.
- 204 - 클라이언트의 요청응 정상적으로 수행하였을때 사용합니다. 200과 다른점은 204는 응답 바디가 없을때 사용합니다. 예를들어 DELETE와 같은 요청시에 사용합니다. 클라이언트의 리소스 삭제요청이 성공했지만 부가적으로 응답 바디에 넣어서 알려줄만한 정보가 하나도 없을땐 204를 사용합니다.

실패

- 400 - 클라이언트의 요청이 부적절할때 사용합니다. 요청 실패시 가장 많이 사용될 상태코드로 예를들어 클라이언트에서 보낸 것들이 서버에서 유효성 검증(validation)을 통과하지 못하였을때 400으로 응답합니다. 응답 바디에 요청이 실패한 이유를 넣어줘야 합니다.
- 401 - 클라이언트가 인증되지 않은 상태에서 보호된 리소스를 요청했을때 사용하는 요청입니다. 예를들어 로그인(login)하지 않은 사용자가 로그인했을때에만 요청 가능한 리소스를 요청했을때 401을 응답합니다.
- 403 - 사용자 인증상태와 관계 없이 응답하고싶지 않은 리소스를 클라이언트가 요청했을때 사용합니다. 그러나 해당 응답코드 대신 400을 사용할 것을 권고합니다. 그 이유는 일단 403 응답이 왔다는것 자체는 해당 리소스가 존재한다는 뜻입니다. 응답하고싶지 않은 리소스는 존재 여부 조차 감추는게 보안상 좋기때문에 403을 응답해야할 요청에 대해선 그냥 400이나 404를 응답하는게 좋겠습니다.
- 404 - 클라이언트가 요청한 리소스가 존재 하지 않을때 사용하는 응답입니다.
- 405 - 클라이언트가 요청한 리소스에서는 사용 불가능한 Method를 이용했을때 사용하는 응답입니다. 예를들어 읽기전용 리소스에 DELETE Method를 사용했을때 405 응답을 하면 됩니다.

기타

- 301 - 클라이언트가 요청한 리소스에 대한 URI가 변경 되었을때 사용합니다. 응답시 `Location` header에 변경된 URI를 적어줘야 합니다.
- 500 - 서버에 뭔가 문제가 있을때 사용합니다.

마크 마세가 쓴 REST API 디자인 규칙에 이런 말이 있네요.

> REST API는 부실한 HTTP 클라이언트에 부합하려는 그 어떤 타협도 해서는 안된다.

마치며

순서가 약간 뒤죽박죽 작성된 느낌이 있네요. 사실 저도 RESTFul한게 좋은건지 나쁜건지 뭔지 아직도 잘 모르는 상태로 작성하였는데, 한가지 확실한건 REST라는 개념(?)이 널리 퍼지고 많은 API들이 RESTFul하다면 REST에서 다루는 내용들은 따로 문서화도 필요없이 자연스레 좀더 체계적인 느낌으로 API를 사용할 수 있을꺼 같은 느낌이 듭니다. 아직 미완성된 글이라 생각합니다. 틀린 내용이나 이해되지 않는 내용이 있으시면 댓글이나 기타 편하신 방법을 통해 적극적으로 글을 완성해주시길 바랍니다.

참고자료

- 일관성 있는 웹 서비스 인터페이스 설계를 위한 REST API 디자인 규칙
- REST 아키텍처를 훌륭하게 적용하기 위한 몇 가지 디자인 팁
- RESTful API를 설계하기 위한 디자인 팁
- 내용협상 (Content Negotiation)

Jan 28th, 2014
김한기
rest
Comments

Comments

8 years ago

글 잘 읽었습니다. 고맙습니다.

0 0 Reply

WooJung Kim

8 years ago

공부하는데 많은 도움이 되었습니다. 좋은글 감사합니다 :)

0 0 Reply

Kim Jun Ho

8 years ago

마지막에
일관성 있는 웹 서비스 인터페이스 설계를 위한 REST API 디자인 규칙
링크 주소가 잘 못 된거 같습니다.

0 0 Reply

김 **김한기** Mod → Kim Jun Ho

8 years ago

지적 감사합니다. 현재 본문 수정에 문제가있어 댓글로 올바른 링크를 알려드립니다.

해당 참고자료의 올바른 링크는 http://www.hanbit.co.kr/sto... 입니다.

0 0 Reply

박찬석

8 years ago

등록/수정 화면 호출에는 어떤 URI를 써야 하나요?

0 0 Reply

jsveron23 → 박찬석

8 years ago

말씀처럼 표준은 없지만,
new 라는 단어는 http method 로 표현이 가능합니다.

즉 /sports 를 post method -> create, (new)
/sports/1 를 get 이면 그냥 웹에 뿌리기
/sports/1 를 put 이면 수정
/sports/1 를 delete 면 삭제

uri 자체에 method 에 해당하는 내용 즉 edit 같은 내용은 넣지 않는게 좋은 듯 합니다.

0 0 Reply

김 **김한기** Mod → jsveron23

8 years ago

박찬석님의 질문은 CRUD 동작에 대한 URI가 아닌,

등록하는것과 수정하는 화면(view)호출에 대한 URI를 질문하시는것 같습니다.

그래서 view 호출 화면에선 new나 edit같은 동사를 사용해야 합니다...

0 0 Reply

jsveron23 → 김한기

8 years ago

제 글이 왜 pending 이 뜰까요;;;

http://www.codeproject.com/...

위 페이지의 윗 부분에 CRUD 주소 설계가 나옵니다.

Create: POST http://localhost:51377/api/values
Read: GET http://localhost:51377/api/values/{id}
Update: PUT http://localhost:51377/api/values/{id}
Delete: DELETE http://localhost:51377/api/values/{id}

동사가 꼭 필요하지는 않을 듯 보여요. 질문하신게 제가 제대로 이해 했다면요.

0 0 Reply

김

김한기 Mod → jsveron23

8 years ago

이유는 모르겠으나 disqus자체적으로 pending을 시켜서 제가 최근에 올리신거만 approve 시켰습니다(두개의 댓글이 같은 내용인거 맞죠?) ^^;

논의하던것으로 돌아가서.

말씀하신대로

Create, Read, Update, Delete등은 모두 공감하고 레일즈에서도 그렇게 채용하고있습니다.

예를 들어서 설명하면,

어떠한 블로그가 있습니다. 블로그에 글을 쓰기위해서는,

글을 작성하기위한 화면에 접근해야합니다(1),
그리고 글을 작성하고,
작성한 글을 서버에 저장하기위해 `publish`나 `완료`버튼을 클릭하여 서버로 저장합니다.(2)

위 흐름에서, 질문자님께서 언급하신것은

(1)을 위한 URI를 뭘로 써야하나 질문하신것으로 보여지고, 이 URI로 rails의 경우엔 `new`라는 동사를 uri에서 사용하고있고요.
그리고 (2)번 상황에서 jsveron23님께서 언급하신대로 POST http://localhost:1234/api/values 를 서버에 http request를 날리면 될꺼같고요.

수정화면도 마찬가지로

수정화면을 호출하는 api와 수정된 결과물을 서버에 날릴때 UPDATE를 날리면 될꺼고요.

0 0 Reply

jsveron23 → 김한기

8 years ago

이 글도 확실히 이해 못하는 점을 보니, 제가 둘다 이해 못한듯 합니다. 제 실수 인듯 합니다. :)

0 0 Reply

김 **김한기** Mod → jsveron23
8 years ago
앞으로도 자주 찾아주셔서 다양하고 좋은 의견 부탁드립니다. 감사합니다!

0 0 Reply

김 **김한기** Mod → 박찬석
8 years ago
본문에도 말씀드린것처럼 이렇다할 표준이 없어, 이게 맞다 라고 이야기할 순 없지만

ruby 프레임워크인 Rails는

이렇게 사용을 하고있습니다.

/sports/new <- 등록하는 화면의 URI

/sports/1/edit <- id 1번인 sport의 수정 화면의 URI

참조 링크: http://guides.rubyonrails.o...

0 0 Reply

Francis Kim
8 years ago
감사합니다!

0 0 Reply

염 **염준호**
8 years ago
아
간만에 너무 좋은 정보 잘읽었습니다.
웨드프레스가 html5+css+script로 간다라는걸보다 넘어왔는대요
정말 잘읽었습니다

0 0 Reply

니 **니트넷**
9 years ago
재미있게 잘 읽었습니다.

0 0 Reply

탈 **탈환대**
9 years ago
RESTFul한게 좋은건지 나쁜건지... restful 한게 뭔가요?

0 0 Reply

김 **김민섭**
9 years ago
좋은 글 감사합니다^^

0 0 Reply

김 **김경태**
9 years ago
좋은 글 감사합니다~
0 0 Reply

 **TaeHee Kim**
9 years ago
좋은 글 감사드립니다 :)
0 0 Reply

 **Minseok Son**
9 years ago
공감을 안할 수 없는 정말 좋은글이네요^^
0 0 Reply

뭉 **뭉치**
10 years ago
감사합니다. 정말 잘 읽고 갑니다 ^^
0 0 Reply

김 **김한기** Mod ↗ 뭉치
10 years ago