🌐 Restful

📌 목차

🤔 REST란?

- REST(Representational State Transfer)는 월드 와이드 웹과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처의 한 형식
  - REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에 웹의 장점을 최대한 활용할 수 있는 아키텍처 스타일
  - REST는 **네트워크 상에서 Client와 Server 사이의 통신 방식 중 하나**이다

> 💡 하이퍼 미디어?
>
> - 문자와 숫자로 이루어진 텍스트 외에도 소리와 그림, 애니메이션 등 다양한 정보 매체를 곧바로 접근할 수 있도록 표현하는 기능

- 자원을 이름으로 구분하여 해당 자원의 상태를 주고 받는 모든 것을 의미
  - 즉, 자원(resource)의 표현(representation)에 의한 상태 전달
  1. 자원의 표현
     - 자원: 해당 소프트웨어가 관리하는 모든 것
     - ex) 문서, 그림, 데이터 등
  2. 상태 전달
     - 데이터가 요청되어지는 시점에서 자원의 상태
     - JSON, XML 등을 이용해 데이터를 주고받는 것이 일반적

💰 REST 구성 요소

1. 자원, URL

- 모든 자원은 고유한 ID를 가지며, ID는 서버에 존재
- 클라이언트는 각 자원의 상태를 조작하기 위해 요청을 보낸다
  - HTTP에서 자원을 구별하는 방법은 `student/1`과 같은 HTTP URL

2. 행위, Method

- 클라이언트는 URL을 이용해 자원을 지정하고 자원을 조작하기 위해 Method를 사용한다
- HTTP Protocol에서는 `GET`, `POST`, `DELETE` 같은 Method를 제공

3. 표현

- 클라이언트가 서버로 요청을 보냈을 때 응답 자원의 상태를 의미
- REST에서 하나의 자원은 JSON, XML 등 여러 형태의 표현으로 나타날 수 있다

👻 REST가 필요한 이유

- REST API의 목적은 **이해하기 쉽고 사용하기 쉬운 API를 제작**하는 데에 존재

1. `GET /deleteUserinfo?id=3`
2. `DELETE /users/3`

- 두 요청은 모두 특정 user의 정보를 삭제하는 API
- 그러나 1번 API는 컨벤션이 일관적이지 않을 수 있으며, 이후 API의 이해도를 떨어트릴 수 있음
- REST API를 사용한다고 해서 성능이 향상되는 것은 아님
- **일관적인 컨벤션과 API 이해도 및 호환성을 높이는 것이 REST API의 목적**

🐿️ REST 특징

1. Uniform

- URI로 지정한 리소스에 대한 조작을 통일되고 한정적인 인터페이스로 수행

2. Stateless

- 작업을 위한 상태정보를 따로 저장하고 관리하지 않음
  - 따라서, 세션 정보나 쿠키 정보를 별도로 저장하고 관리하지 않음
- API 서버는 들어오는 요청만을 단순히 처리하면 된다
- 따라서, 서비스의 자유도가 높아짐

3. Cacheable

- REST는 HTTP 웹표준을 그대로 사용하기 때문에 웹에서 사용하는 기존 인프라를 그대로 활용하는 것이 가능
- 따라서 HTTP가 가진 캐싱 기능 적용이 가능

> 💡 HTTP의 캐싱 기능이란?
>
> - HTTP 응답을 저장해둔다
> - 다음에 동일한 HTTP 요청이 시도되면 저장해 둔 HTTP 응답을 재활용 한다

4. Self-descriptiveness

- REST API 메시지만 보고도 동작을 쉽게 이해할 수 있는 구조로 되어있음

5. Client-Server 구조

- REST 서버는 API를 제공하는 역할
- 클라이언트는 사용자 인증이나 세션 등을 직접 관리하는 구조로 각각의 역할이 확실히 구분됨
- 클라이언트와 서버간의 의존성이 줄어들게 된다

6. 계층형 구조

- REST 서버는 다중 계층으로 구성될 수 있다
- 보안, 로드 밸런싱, 암호화 계층을 추가해 구조상의 유연성을 둘 수 있음

🦊 RESTful

- REST 아키텍처를 구현하는 웹 서비스를 나타내기 위해 사용되는 용어
- REST 원리를 따르는 시스템

RESTful API 특징

- 확장성과 재사용성을 높여 유지보수 및 운영을 편리하게 한다
- HTTP 표준을 기반으로 구현하므로, HTTP를 지원하는 프로그램 언어로 클라이언트, 서버를 구축할 수 있다

RESTful API 작성하는 법

1. URI는 정보의 자원을 표현해야 한다
    - resource는 동사보다 **명사**를, 대문자보다 **소문자**를 사용한다
    - resource의 도큐먼트 이름으로는 **단수 명사**를 사용해야 한다
    - resource의 컬렉션 이름으로는 **복수 명사**를 사용해야 한다
        - 컬렉션은 객체의 집합, 도큐먼트는 객체라고 이해하면 됨

```
GET /members/1
```

2. 자원에 대한 행위는 HTTP Method로 표현한다
3. URI에 HTTP Method가 들어가면 안된다

```
GET /books/delete/1 (x)
DELETE books/1 (o)
```

4. URI 행위에 대한 동사 표현이 들어가면 안된다

```
GET /books/show/1 (x)
GET /books/1 (o)
```

5. id는 하나의 특정 resource를 나타내는 고유값을 의미한다
6. `/`는 계층 관계를 나타내는데 사용한다
7. URI 마지막 문자로 `/`를 포함하지 않는다
8. URI에 포함되는 모든 글자는 resource의 유일한 식별자로 사용되어야 한다
    - URI가 다르다는 것은 resource가 다르다는 것이다
    - resource가 다르면 URI도 달라져야 한다
9. `-`은 URI 가독성을 높이는데 사용할 수 있다
10. `_`은 URI에 사용하지 않는다
11. URI 경로에는 소문자가 적합하다
12. 파일 확장자는 URI에 포함하지 않는다
13. 리소스 간에 연관 관계가 있는 경우 다음과 같은 방법으로 표현한다
    - `/리소스명/리소스 ID/관계가 있는 다른 리소스 명`

```
GET /books/{bookid}/viewers (일반적으로 소유 'has'의 관계를 표현할 때)
```

🥊 HTTP Methods

- RESTful API를 위한 HTTP Methods

1. GET

- **자원을 받아오기만 할 때 사용**
- 어떤 방식으로도 자원의 상태를 변경시키지 않으므로 safe method라고 불리기도 함
- GET API는 멱등성을 띈다

> 💡 멱등성?
>
> - 동일한 API를 여러번 호출할 때에도 동일한 결과를 얻을 수 있음을 의미
> - 단, POST, PUT을 통해 데이터가 변경되지 않을 시

2. POST

- **새로운 자원을 추가할 때 사용**
- 서버의 상태를 변경시킨다
  - 따라서 비멱등성 성질을 가짐
- 응답 코드로 `201(Created)`를 받아야 정상적으로 서버에 추가되었음을 확인할 수 있음

3. PUT

- **존재하는 자원을 변경할 때 사용**
- 만약 자원이 존재하지 않는 경우 API는 새로운 자원을 생성하도록 할 수 있다
- 멱등성을 가진다

> PUT 동작 방식
>
> if resource 존재 then update()
>
> else then create()

> 💡 POST vs PUT

	POST	PUT
Resource identifier 유무	X	O
멱등성 여부	X	O

4. DELETE

- **자원을 삭제할 때 사용**
- DELETE 메소드는 멱등성 성질을 띈다

5. PATCH

- **한 자원의 데이터를 부분적으로 변경할 때 사용**
  - PUT도 마찬가지로 자원을 변경할 수 있으나, 존재하는 자원에 대해 부분적으로 업데이트를 할 때는 PATCH를 이용한다
  - PUT은 자원을 완전히 대체하는 경우 사용
- PATCH의 경우 모든 브라우저, 서버, 앱 어플리케이션 프레임워크에서 사용할 수 있는 것은 아님

> 💡 PATCH vs PUT
>
> - PUT 요청 시 요청을 일부분만 보낸 경우 나머지는 default 값으로 수정되는 것이 원칙
> - 따라서, 바뀌지 않는 속성도 모두 보내야 한다
>   - 만약 전체가 아닌 일부분만 전달할 경우 전달한 필드 외의 다른 속성은 모두 null 혹은 default 값 처리가 되니 주의

🔗 참고

▼ 링크

RESTful API POST와 PUT의 차이 · Studio u by kingjakeu

흔히들 POST는 자원을 생성하고, PUT은 생성 혹은 갱신하는 것이 라고 한다.
결국 둘 다 자원을 생성하는 거라면, POST와 PUT의 차이는 무엇인지 알아보자.

😊 https://kingjakeu.github.io/study/2020/07/15/http-post-put/

[REST API] REST API 규칙/PUT과 POST 차이/PUT과 PATCH 차이

먼저, REST란? Representational State Transfer의 약자이며, 다음과 같이 구성되어 있다. 자원(Resource): URI 행위(Verb): HTTP Method 표현(Representations) 즉 REST는 URI를 통해 자원을 표시하고, HTTP Method

🥑 https://devuna.tistory.com/77