
spring

REST

- REST(Representational State Transfer)의 약자로 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 모든 것을 의미한다.

- REST는 클라이언트와 서버 사이의 통신 방식 중 하나이고, 웹의 기존 기술과 HTTP 프로토콜을 그대로 사용할 수 있는 아키텍처이다.

- 즉, HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고,
  HTTP Method(POST, GET, PUT, DELETE, PATCH 등)를 통해
  해당 자원(URI)에 대한 CRUD Operation을 적용하는 것을 의미한다.

REST 구성 요소

- REST는 다음과 같은 3가지로 구성이 되어있다.

  자원(Resource) : HTTP URI

  자원에 대한 행위(Verb) : HTTP Method

  자원에 대한 행위의 내용[표현] (Representations) : HTTP Message Pay Load(요청에 대한 적절한 응답)

REST 특징

- Server-Client(서버-클라이언트 구조) : REST Server와 Client 간의 의존성이 줄어든다.
- Stateless(무상태) : HTTP 프로토콜이 Stateless Protocol이므로 REST도 무상태성을 갖는다.
- Cacheable(캐시 처리 가능) : HTTP가 가진 캐싱 기능을 적용할 수 있어서 대량의 요청을 효율적으로 처리 가능하다.
- Layered System(계층화) : 다중 계층으로 구성하여, 구조 상의 유연성을 줄 수 있다.
- Uniform Interface(인터페이스 일관성) : URI로 지정한 Resource에 대한 조작을 통일되고 한정적인 인터페이스로 수행한다.

REST 장점 & 단점

- HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구축할 필요가 없다.
- HTTP 프로토콜의 표준을 최대한 활용하여 여러 추가적인 장점을 함께 가져갈 수 있게 해준다.
- HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다.
- Hypermedia API의 기본을 충실히 지키면서 범용성을 보장한다.
- REST API 메시지가 의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악할 수 있다.
- 여러 가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.
- 서버와 클라이언트의 역할을 명확하게 분리한다.

- 표준이 자체가 존재하지 않아 정의가 필요하다.
- HTTP Method 형태가 제한적이다.
- 브라우저를 통해 테스트할 일이 많은 서비스라면 쉽게 고칠 수 있는 URL보다 Header 정보의 값을 처리해야 하므로 전문성이 요구된다.

REST API (Representational State Transfer API)

- RESPT API란 REST의 원리를 따르는 API를 의미한다.

- 올바른 REST API 설계를 위해서는 몇 가지 규칙을 따라야 한다.

1. URI는 동사보다는 명사를, 대문자보다는 소문자를 사용하여야 한다.
2. 마지막에 슬래시 (/)를 포함하지 않는다.
3. 언더바 대신 하이폰을 사용한다.
4. 파일확장자는 URI에 포함시키지 않는다.
5. 행위를 포함하지 않는다.

- REST의 원리를 따르는 시스템을 "RESTful 하다" 라고 한다.