API (Application Programming Interface)란?

- API는 응용 프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻한다.

API의 유형

1) private API

: 회사 개발자가 자체 제품과 서비스를 개선하기 위해 내부적으로 발해하는 API. 제 3자에게 노출되지 않는다.

2) public API

: 누구나 제한 없이 API를 사용할 수 있는 개방형 API

3) partner API

: 기업이 데이터 공유에 동의하는 특정인들만 사용할 수 있는 API. 비즈니스 관계에서 사용되는 편이며, 종종 파트너 회사 간의 소프트웨어를 통합하기 위해 사용된다.

Rest란?

: REST는 "REpresentational State Transfer"의 약자로, Roy Fielding의 박사학위 논문에서 최초로 소개되었다.

: REST는 HTTP를 기반으로 필요한 자원에 접근하는 방식을 정해놓은 아키텍처라고 할 수 있다. 여기서 자원이란 소프트웨어가 관리하는 모든 것(문서, 그림, 데이터 등)을 의미한다.

: REST는 월드 와이드 웹(www)과 같은 분산 하이퍼 미디어 시스템을 위한 소프트웨어 개발 아키텍처의 한 형식이다.

: REST는 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에 웹의 장점을 최대한 활용할 수 있다.

REST의 속성

1) 서버에 있는 모든 자원은 각 자원 당 클라이언트가 바로 접근할 수 있는 고유 URI가 존재한다.

2) 모든 요청은 클라이언트가 요청할 때마다 필요한 정보를 주기 때문에 서버에서 세션 정보를 보관할 필요가 없다. 따라서 서비스에 자유도가 높아지고 유연한 아키텍처 적용이 가능하다.

3) HTTP 메서드를 사용한다. 모든 자원은 HTTP 인터페이스인 GET, POST, PUT, DELETE 등의 메서드로 접근되어야 한다.

4) 서비스 내에 하나의 자원이 주변에 연관된 자원들과 연결되어 표현이 되어야 한다.

REST의 구성 요소

1) 자원(Resource)

모든 자원에는 고유한 ID가 존재하고, 이 자원은 서버에 존재한다. REST는 자원에 접근할 때 URI(Uniform Resource Identifier)를 이용한다. 여기서 URI는 자원의 위치를 나타내는 일종의 식별자이다.

2) 메서드(Method)

REST는 기본적으로 HTTP 메서드를 사용한다. 그 종류에는 GET, POST, PUT, DELETE 등이 존재하고, 각각 다음과 같은 역할을 한다.

GET : 해당 리소스를 조회한다.

POST : 해당 리소스를 생성한다

PUT : 해당 리소스를 수정한다.

DELETE : 해당 리소스를 삭제한다.

위와 같은 연산을 CRUD(Create, Research, Update, Delete) 연산이라고 한다.

3) 메시지(Message)

: 메시지는 HTTP header, body, 응답 상태 코드 등으로 구성되어 있다. 간단하게 설명하자면 header에는 body에 어떤 형식으로 데이터가 담겼는지 표시하고, body는 자원에 대한 정보를 JSON, XML 등으로 전달한다. 응답 상태 코드는 200~500 사이의 숫자로 클라이언트의 요청에 대한 상태를 나타내 준다.

REST의 특징

1. Server-Client(서버-클라이언트) 구조

자원이 있는 쪽이 서버, 자원을 요청하는 쪽이 클라이언트가 된다. 서버는 API를 제공하고 비즈니스 로직 처리 및 저장을 책임진다.

2. Stateless(무상태)

HTTP 프로토콜은 기본적으로 무상태이다. REST 역시 HTTP를 기본으로 하기 때문에 무상태이다. 여기서 무상태란 클라이언트의 상태(State)를 서버에 저장하지 않는다는 뜻이다. 따라서 서버는 클라이언트의 요청을 완전히 별개의 것으로 인식하고 처리한다. 따라서 이전의 요청이 다음의 요청에 연관되지 않는다. 이를 통해 서버의 처리 방식에 일관성을 부여하고 부담이 줄어들게 된다.

3. Cacheable(캐시 처리 가능)

HTTP의 캐싱 기능을 적용할 수 있다. 즉, 대량의 요청을 효율적으로 처리하기 위한 캐시를 사용할 수 있다. 캐시 사용을 통해 응답 시간이 빨라지고 성능, 서버의 자원 이용률을 향상할 수 있다.

4. Layered System(계층화)

클라이언트는 REST API 서버만 호출한다. REST 서버는 다중 계층으로 구성될 수 있다. API 서버는 순수 비즈니스 로직을 수행하고 그 앞단에 보안, 로드 밸런싱, 암호화, 사용자 인증 등을 추가하여 구조상의 유연성을 줄 수 있다.

5. Code-On-Demand

서버로부터 스크립트를 받아서 클라이언트에서 실행한다. 이 특징은 반드시 충족할 필요는 없다.

6. Uniform Interface(인터페이스 일관성)

URI로 지정한 자원에 대한 조작을 통일되고 한정적인 인터페이스로 수행한다. HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다. 따라서 특정 언어나 기술에 종속되지 않는다.

REST의 장단점

장점

1. 언어와 플랫폼에 독립적이다.

2. REST API 메시지가 의도하는 바를 명확하게 나타내 의도를 쉽게 파악할 수 있다.

3. REST가 지원하는 프레임워크나 언어 등 도구들이 없어도 구현 가능하다.

4. HTTP를 사용하므로 기존 웹 인프라를 사용 가능하다.

5. 서버와 클라이언트의 역할을 명확하게 분리한다.

6. 여러 가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.

단점

1. HTTP 프로토콜만 사용 가능하다.

2. p2p 통신 모델을 가정하여 둘 이상을 대상으로 하는 분산 환경에 유용하지 않다.

3. 보안, 정책 등에 대한 표준이 없다.

REST API란?

- REST API는 결국 REST를 기반으로 서비스 API를 구현한 것이다. 예를 들어 네이버에서 제공하는 Open API를 이용한다고 하자. 특정 기술, 예를 들어 네이버 지도의 API를 사용하고 싶다면 이때 REST를 이용해 사용하는 것이 REST API다. URI 형식으로 HTTP 메서드(GET, POST, PUT, DELETE)를 요청해 자원을 조회, 생성, 수정, 삭제할 수 있는 것이 REST API인 것이다.

출처

- [Web] API란? REST API란? RESTful 이란? REST 구성 요소, 특징, 장단점 (tistory.com)