**REST API**

REST는 representational state transfer의 약자로 **데이터 자원을 이름으로 구분하여 해당 자원의 상태를 주고 받는 것**을 의미합니다. **rest API는 REST 기반으로 서비스 API를 구현한 것**입니다.

**API**: 응용프로그램 사이에서 통신할 수 있게 인터페이스를 개발하는 것

### 구성 요소

* **자원**은 **URI로 자원**을 명시
* 메소드(**행위**)는 **POST, GET, PATCH, PUT, DELETE 등의 메소드(CRUD Operation)를 통해 해당 자원에 대한 CRUD Operation을 적용**합니다.
  + CRUD란 Create, Read, Update, Delete란 뜻
* 자원에 대한 행위의 내용(**http message pay load**)\*\*으로 자원의 상태에 대한 조작을 요청하면 서버는 JSON, XML, TEXT, RSS 등으로 응답을 보냅니다.

### 장점

* REST API 메시지가 **의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악**할 수 있습니다.
* 서버와 클라이언트의 역할을 명확하게 분리합니다.
* HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라 구축이 필요없습니다.

### 단점

* 브라우저를 통해 테스트할 일이 많은 서비스라면 쉽게 고칠 수 있는 URI보다 Header 정보의 값을 처리해야하므로 전문성이 요구됩니다.

### 특징

* **Stateless**: Client의 context를 Server에 저장 X. 즉 세션과 쿠키와 같은 context 정보를 신경쓰지 않아도 되므로 구현이 단순해집니다.
  + **Server-Client**: 자원이 있는 쪽이 Server, 자원을 요청하는 쪽이 Client가 됩니다.
* **Cacheable**: 웹 표준 HTTP 프로토콜을 그대로 사용하므로 웹에서 사용하는 기존 인프라를 그대로 활용 가능. 캐싱 기능 적용 가능
* **Layered System**: Client는 REST API Server만 호출, REST SERVER은 다중 계층으로 구성 가능, API 서버는 순수 비즈니스 로직만 수행하고 그 앞단에 보안, 로드밸런싱, 암호화, 사용자 인증 등을 추가하여 구조상의 유연성을 줄 수 있습니다.
* **Uniform Interface**: URI로 지정한 자원에 대한 조작을 통일되고 한정적인 인터페이스로 수행한다. 따라서 HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용 가능하고 특정 언어나 기술에 종속되지 않습니다.

**HTTP 통신**

브라우저와 서버가 통신할 수 있도록 만들어주는 여러 프로토콜 가운데 한 종류로 **웹 브라우저**와 **웹 서버** 사이에 데이터를 주고받는데 쓰이는 응용계층의 통신 프로토콜입니다.

**웹페이지 검색 원리**

크게는 클라이언트가 서버에 요청을 보내면 서버는 해당 요청에 대해 응답하는 것입니다.

1. 사용자가 웹 브라우저의 검색창에 특정 사이트를 입력합니다.
2. 웹 브라우저가 DNS 서버에서 특정 사이트의 주소를 요청하고 IP주소를 응답으로 받습니다.
3. 3 way handshake를 통해 서버와 클라이언트 사이에 데이터를 송수신할 준비를 합니다.
4. https의 경우 TLS handshake 과정을 추가로 거칩니다.
5. 웹 브라우저가 웹 서버에게 해당 IP로 html 문서를 요청합니다.
6. 웹 서버는 WAS와 DB에서 웹페이지 작업을 처리한 후에 결과를 웹 브라우저에게 html 문서로 응답합니다.
7. 웹 브라우저는 화면에 웹 페이지를 출력합니다.