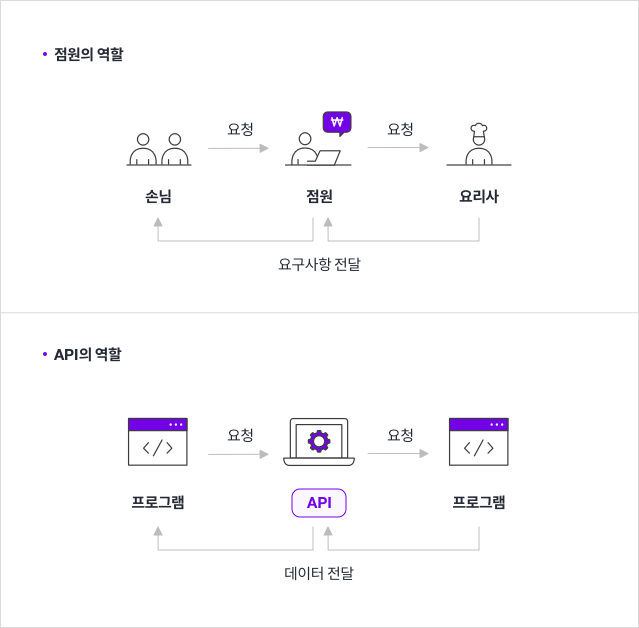
**1. 서버API 통신이란 무엇이고, REST API란 무엇인가.**



점원은 손님에게 메뉴를 알려주고, 주방에 주문받은 요리를 요청합니다. 그다음 주방에서 완성된 요리를 손님께 다시 전달하지요. API는 점원과 같은 역할을 합니다.

API는 손님(프로그램)이 주문할 수 있게 메뉴(명령 목록)를 정리하고, 주문(명령)을 받으면 요리사(응용프로그램)와 상호작용하여 요청된 메뉴(명령에 대한 값)를 전달합니다.

쉽게 말해, **API는 프로그램들이 서로 상호작용하는 것을 도와주는 매개체**로 볼 수 있습니다.

**1. API는 서버와 데이터베이스에 대한 출입구 역할을 한다.**

: 데이터베이스에는 소중한 정보들이 저장되는데요. 모든 사람들이 이 데이터베이스에 접근할 수 있으면 안 되겠지요. API는 이를 방지하기 위해 여러분이 가진 서버와 데이터베이스에 대한 출입구 역할을 하며, 허용된 사람들에게만 접근성을 부여해줍니다.

**2. API는 애플리케이션과 기기가 원활하게 통신할 수 있도록 한다.**

: 여기서 애플리케이션이란 우리가 흔히 알고 있는 스마트폰 어플이나 프로그램을 말합니다. API는 애플리케이션과 기기가 데이터를 원활히 주고받을 수 있도록 돕는 역할을 합니다.

**3. API는 모든 접속을 표준화한다.**

API는 모든 접속을 표준화하기 때문에 기계/ 운영체제 등과 상관없이 누구나 동일한 액세스를 얻을 수 있습니다. 쉽게 말해, API는 범용 플러그처럼 작동한다고 볼 수 있습니다.

**REST(Representational State Transfer)**는 **네트워크를 통해서 컴퓨터들끼리 통신할 수 있게 해주는 아키텍처 스타일**입니다.

자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미한다.즉, 자원(resource)의 표현(representation) 에 의한 상태 전달이 REST라고 할수 있다. 데이터가 요청되어지는 시점에서 자원의 상태(정보)를 전달한다. JSON 혹은 XML를 통해 데이터를 주고 받는 것이 일반적이다. 월드 와이드 웹(www)과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 개발 아키텍처의 한 형식. REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에 웹의 장점을 최대한 활용할 수 있는 아키텍처 스타일이다. REST는 네트워크 상에서 Client와 Server 사이의 통신 방식 중 하나이다.

**REST의 구체적인 개념**

HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고, HTTP Method(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미한다.  
즉, REST는 자원 기반의 구조(ROA, Resource Oriented Architecture) 설계의 중심에 Resource가 있고 HTTP Method를 통해 Resource를 처리하도록 설계된 아키텍쳐를 의미한다.  
웹 사이트의 이미지, 텍스트, DB 내용 등의 모든 자원에 고유한 ID인 HTTP URI를 부여한다.

**REST의 장점**

HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구출할 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.

여러가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.

서버와 클라이언트의 역할을 명확하게 분리한다.

**REST의 단점**

표준이 존재하지 않는다.

사용할 수 있는 메소드가 4가지 밖에 없다.

HTTP Method 형태가 제한적이다.

브라우저를 통해 테스트할 일이 많은 서비스라면 쉽게 고칠 수 있는 URL보다 Header 값이 왠지 더 어렵게 느껴진다.

구형 브라우저가 아직 제대로 지원해주지 못하는 부분이 존재한다.

PUT, DELETE를 사용하지 못하는 점

pushState를 지원하지 않는 점

**REST가 필요한 이유**

‘애플리케이션 분리 및 통합’

‘다양한 클라이언트의 등장’

최근의 서버 프로그램은 다양한 브라우저와 안드로이폰, 아이폰과 같은 모바일 디바이스에서도 통신을 할 수 있어야 한다.

이러한 멀티 플랫폼에 대한 지원을 위해 서비스 자원에 대한 아키텍처를 세우고 이용하는 방법을 모색한 결과, REST에 관심을 가지게 되었다.

**REST API의 정의**

REST 기반으로 서비스 API를 구현한 것

사내 시스템들도 REST 기반으로 시스템을 분산해 확장성과 재사용성을 높여 유지보수 및 운용을 편리하게 할 수 있다. REST는 HTTP 표준을 기반으로 구현하므로, HTTP를 지원하는 프로그램 언어로 클라이언트, 서버를 구현할 수 있다. 즉, REST API를 제작하면 델파이 클라이언트 뿐 아니라, 자바, C#, 웹 등을 이용해 클라이언트를 제작할 수 있다.

**RESTful이란**

RESTful은 일반적으로 REST라는 아키텍처를 구현하는 웹 서비스를 나타내기 위해 사용되는 용어이다. ‘REST API’를 제공하는 웹 서비스를 ‘RESTful’하다고 할 수 있다. RESTful은 REST를 REST답게 쓰기 위한 방법으로, 누군가가 공식적으로 발표한 것이 아니다. 즉, REST 원리를 따르는 시스템은 RESTful이란 용어로 지칭된다.

**RESTful의 목적**

이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것

RESTful한 API를 구현하는 근본적인 목적이 성능 향상에 있는 것이 아니라 일관적인 컨벤션을 통한 API의 이해도 및 호환성을 높이는 것이 주 동기이니, 성능이 중요한 상황에서는 굳이 RESTful한 API를 구현할 필요는 없다.

**RESTful 하지 못한 경우**

Ex1) CRUD 기능을 모두 POST로만 처리하는 API

Ex2) route에 resource, id 외의 정보가 들어가는 경우(/students/updateName)