인터넷에서 사용되는 여러 기술 : REST API 1

조인준
KBS 미디어기술연구소 차장

지난 편들을 통해서 HTTP 프로토콜에 관한 개요를 살펴보았습니다. 이번 편에서는 HTTP와 연관되어 많이 들어보셨을 REST API에 관해 다루어보겠습니다.

많은 자료에서 REST API는 '**RE**presentational **S**tate **T**ransfer **A**pplication **P**rogramming **I**nterface의 약자로 웹 서비스 간 데이터를 주고받을 수 있도록 설계된 소프트웨어 인터페이스' 정도로 소개되고 있습니다. 솔직히 고백하면 C군은 이 소개 내용을 전혀 이해하지 못했습니다. 그냥 그런가 보다 하는 느낌으로 모르면 모르는 대로 그냥 사는데 큰 문제가 없으니 넘어갔습니다. 하지만 이제는 누군가에게 설명을 해야 하는 입장이므로 사명감을 갖고 방대한 정보의 바다, 인터넷을 뒤져 도대체 'Representational State Transfer Application Programming Interface'의 의미가 무엇인지 진정성을 갖고 치열하게 찾아보았고 몇몇 양질의 자료를 발견한 후 이들을 상호 비교 및 대조하여 다음과 같이 정리했습니다. 단계적 설명을 위해 REST(**RE**presentational **S**tate **T**ransfer)와 API(**A**pplication **P**rogramming **I**nterface)를 나누어 설명하겠습니다. API는 독자 여러분도 여러 분야에서 지속적으로 들어와서 아마 REST보다 친숙할 것 같습니다.

REST (REpresentational State Transfer)

HTTP는 클라이언트/서버 모델을 기반으로 동작합니다. 웹브라우저 등의 클라이언트를 통해 웹페이지의 HTML이나 각종 데이터 등을 서버에 요청하고, 서버는 해당 요청을 수신한 후 요청에 대한 처리와 응답을 생성하여 클라이언트에게 회신하는 구조를 바탕으로 동작합니다.

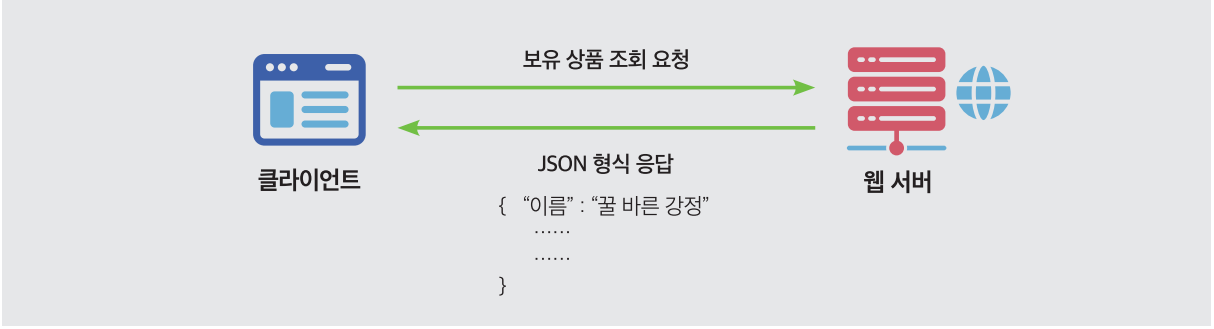

그림 1. 인터넷을 통한 조회 요청 및 응답

[그림 1]을 통해 Representational State Transfer의 의미를 분석해보겠습니다. [그림 1]은 인터넷 상점의 보유 상품을 조회하는 상황을 가정했습니다. 클라이언트는 상점이 보유한 상품 리스트에 대한 요청(Request)을 보냅니다. 이 요청에 대해 서버는 JSON 등의 서식을 통해 [표 1]과 같이 보유한 상품의 리스트를 응답(Response)으로 보냅니다.

우선 Representational은 [표 1]과 같은 Representation(표현)을 의미합니다. 도대체 무슨 소리인가 싶을 것 같습니다. C군 스스로도 설명하기가 조금 애매합니다만 중언부언이라도 좀 더 열심히 표현해본다면, [표 1]의 JSON은 과자나 면 등 상품의 종류에 따라서 상점이 보유한 상품 및 현재 재고 수량을 나타내고 있습니다. 이는 상품에 관한 정보를 관리하고 제공하는 서버의 '리소스'를 JSON 포맷으로 Representation 한 것입니다. 두 번째로 State는 [표 1]과 같이 상품 등의 상태를 의미합니다. 과자나 면같이 어느 종류의 상품을 보유하고 있는지, 종류마다 어떤 상품을 보유하고 있는지, 각 상품의 재고는 얼마나 되는지 등이 State에 해당합니다. 마지막으로 Transfer는 서버의 리소스의 State(상태)를 JSON으로 Representation(표현) 한 것을 서버, 클라이언트가 서로 전달한다는 의미입니다. 이제 이 모두를 합친 Representational State Transfer의 의미가 어느 정도 감이 오는 것 같지 않나요? 이야기를 더 진행하기 전에 웹에서 이야기하는 '리소스'라는 용어에 대해 한 번 정리하는 것이 좋을 것 같습니다.

```
{ "과자" : [
            {"이름" : "꿀 바른 강정"
             "수량" : 20},

            {"이름" : "꿀물 가득 샌드"
             "수량" : 30}
             ......
          ],

  "면류" : [
            {"이름" : "불갈비 범벅면"
             "수량" : 15},
             ......
          ],
  .....
}
```

표 1. JSON 포맷의 서버 응답 예시

- **리소스(Resource)**
  - 웹에서 '리소스'는 웹상에서 식별 가능하고 요청 가능한 정보 또는 객체를 나타내며 정보를 표현하고 공유하는 데 사용됩니다.
  - 리소스는 웹상에서 고유하게 식별되어야 하고 이를 위해 URI(Uniform Resource Identifier)를 사용합니다. URI는 리소스를 찾고 요청할 수 있는 일종의 주소와 같은 것이며, 웹페이지, 이미지 파일, 동영상 등은 모두 고유한 URI를 가집니다.
  - 웹에서 리소스는 다양한 형식을 가질 수 있습니다. HTML 문서, 이미지, 텍스트 파일, JSON 또는 XML 문서, 웹 서비스 API 엔드 포인트, 데이터베이스 레코드 등 모든 종류의 데이터와 서비스가 리소스로 표현될 수 있습니다.
  - 웹에서 리소스는 HTTP(Hypertext Transfer Protocol) 요청 및 응답을 통해 전송될 수 있습니다. 클라이언트는 HTTP로 URI를 특정하여 리소스를 요청하고, 서버는 해당 리소스를 HTTP 응답을 통해 클라이언트에게 제공합니다. 예를 들면 웹 브라우저는 웹페이지 리소스를 요청하고 웹 서버는 해당 페이지의 HTML 문서를 HTTP 응답으로 전송할 수 있습니다.

요약하면, 웹에서 리소스는 데이터와 서비스를 표현하고 접근할 수 있는 개체를 나타내며 URI를 기반으로 식별되어 HTTP를 통해 클라이언트와 서버 간에 교환됩니다. 또한, 웹 서비스 및 웹 애플리케이션을 개발하는 데 중요한 역할을 합니다.

API (Application Programming Interface)

Representational State Transfer까지의 의미를 정리하였으니 이제 API가 무엇인지에 대해 조금 더 자세히 알면 이 둘을 합친 REpresentational State Transfer API, 즉 REST API의 실체에 대한 구체적 이해의 기반을 다질 수 있을 것 같습니다.

API는 응용 프로그램 간 정보를 공유하고 상호 작용하기 위한 규칙과 프로토콜의 모음이라고 정의되어 있으며, 이 정의와 같이 응용 프로그램 간에 데이터 및 서비스에 대한 요청 및 응답을 주고받을 수 있도록 하는 인터페이스를 제공합니다.

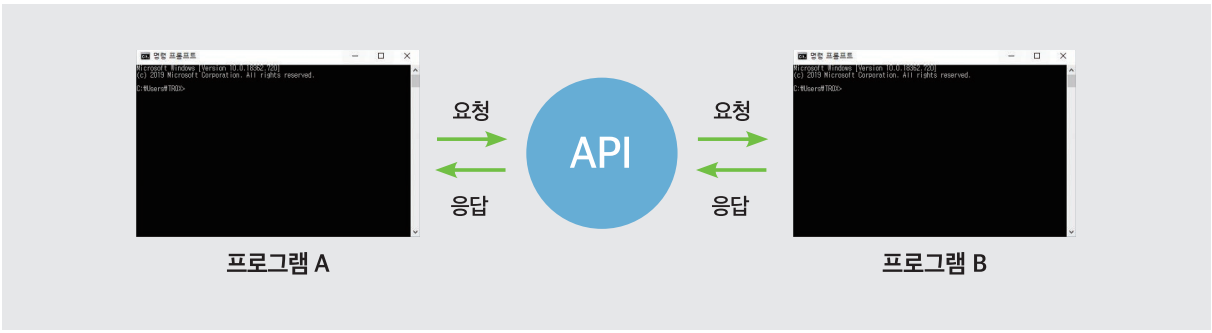

그림 2. API

REST API

이제 앞에서 설명한 REST와 API를 합치면 [그림 3]과 같은 REST API 그림이 됩니다. [그림 2]의 프로그램들이 서버와 클라이언트로 바뀌었고, REST API가 서버와 클라이언트 간에 데이터 및 서비스에 대한 요청 및 응답을 주고받을 수 있도록 하는 인터페이스를 제공합니다. 앞서 API의 핵심은 응용 프로그램 간 요청과 응답 인터페이스라고 했는데, REST API는 어떻게 서버와 클라이언트 사이에 요청과 응답을 주고받는 인터페이스를 제공할까요?

이는 HTTP를 통해 클라이언트와 서버 간 데이터를 교환함으로써 가능해집니다. 지난 편에서 소개드린 것처럼 HTTP는 클라이언트의 요청과 서버의 응답으로 이루어져 있습니다. REST API는 HTTP를 통해 클라이언트가 서버 데이터에 접근할 수 있는 GET, POST, PUT, DELETE 등의 작업 요청 메서드들을 정의합니다. 각 요청을 간단한 예로 설명하면 GET은 서버에게 [표 1]의 예와 같이 리소스를 요청할 때 사용됩니다. POST는 클라이언트가 서버에 새로운 리소스를 등록(예: 인터넷 주문 시 배송지 정보 등)할 때 사용하며, PUT은 클라이언트가 서버에 새로운 리소스를 생성하거나 기존 리소스의 갱신(예: 인터넷 주문 시 배송지 정보 수정 등)을 요청할 때 사용합니다. 마지막으로 DELETE는 리소스 삭제(예: 회원 탈퇴 시 개인정보 삭제)를 요청할 때 사용합니다. GET, PUT, DELETE 등의 요청을 받으면 서버는 이를 처리하고 클라이언트에게 응답을 줍니다.

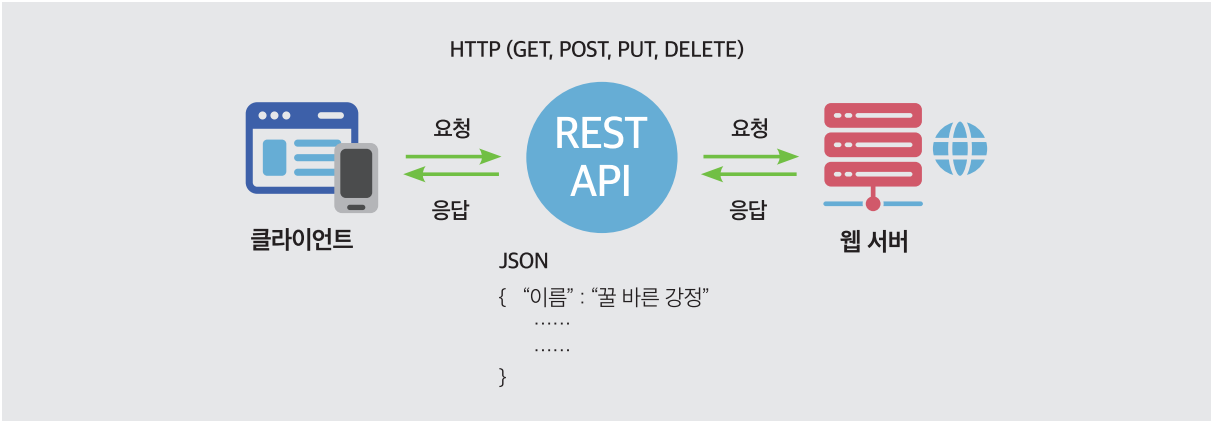

그림 3. REST API

REST는 표준이나 규약이 아닌 아키텍처라고 정의되어있습니다. 이 아키텍처 스타일을 준수하는 웹서비스 또는 API를 RESTful하다고 하며 다음과 같은 특징들이 있습니다.

- **클라이언트-서버 구조 :** REST API는 클라이언트와 서버 간 역할이 분리되어 있어 클라이언트와 서버가 독립적 개발이나 수정을 통해 각자 발전할 수 있으며, 상호 의존성을 낮출 수 있습니다.

• **HTTP 메서드 사용 :** 앞의 설명과 같이 GET, POST, PUT, DELETE 등의 HTTP 메서드를 통해 리소스의 조회 및 등록, 수정, 삭제 등을 지원하며 처리결과를 응답합니다.

• **리소스 중심 :** REST API는 리소스를 중심으로 설계됩니다. 각 리소스는 고유한 URI(Uniform Resource Identifier)를 가지며, 이를 통해 식별됩니다. 예를 들어, [표 1]의 웹 서비스에서 '상품 리스트' 등과 같은 자원은 각각의 URI를 가지고 관리됩니다.

• **Statelessness :** Statelessness는 REST의 중요 아키텍처입니다. 그런데 여기서 뭔가 혼동되기 시작하시죠? REST가 REpresentational State Transfer라고 말해놓고는 이제 와서 Stateless라고 말하는 건 또 뭔가? C군이 지금 자기가 무슨 말을 하는지도 모르고 생각나는 대로 설명이랍시고 써 내려가는 건가? 이런 의문이 독자 여러분의 마음에 눈덩이처럼 커지고 있을 것 같습니다. REpresentational State Transfer의 State는 리소스의 상태를 의미하는 State이고, 지금 이야기하는 Stateless의 State는 서버와 클라이언트들과의 사이에 있어 서버가 클라이언트 관련 데이터를 저장, 추적 및 관리하지 않음을 의미합니다. 이는 클라이언트의 각 요청은 그 자체로 서버가 적절히 처리하기에 충분한 정보를 포함하고 있어야 하며, 서버는 클라이언트의 요청 등에 관한 State(상태)를 추적 및 관리하지 않습니다. 이를 통해 서비스의 확장성을 높일 수 있습니다.

• **캐싱(Caching) :** REST는 HTTP의 캐싱을 활용하여 서버 응답을 클라이언트에서 캐시 할 수 있습니다. 이는 네트워크 부하를 줄이고 응답 성능을 향상시킬 수 있습니다.

• **계층 구조(Layered System) :** REST 아키텍처는 계층적인 구조를 허용합니다. 이는 서버와 클라이언트 간에 중간 계층인 로드 밸런서, 캐싱, 프록시 등을 둘 수 있게 하며, 각 계층은 독립적으로 변경될 수 있습니다.

• **유니폼 인터페이스(Uniform Interface) :** REST API는 일관된 인터페이스를 제공하여 다양한 클라이언트가 이해하고 사용할 수 있게 합니다. 이러한 일관성은 API의 직관성과 사용 편의성을 높입니다.

• **Code on Demand :** 클라이언트가 서버로부터 실행 가능한 코드를 동적으로 다운로드하고 실행할 수 있는 기능을 제공합니다. 이는 서버가 클라이언트에게 스크립트나 프로그램 코드를 제공하는 것을 의미하며 가장 일반적 예는 웹 페이지에서 JavaScript 코드를 다운로드하여 실행하는 것입니다. 웹 브라우저가 웹 페이지를 열면 서버에서 JavaScript 코드를 다운로드하고 실행하여 웹 페이지를 동적으로 변경할 수 있습니다. 이를 통해 웹 애플리케이션의 동적 기능 확장이 용이해집니다. Code on Demand는 REST의 선택적인 조건 중 하나이며, 모든 RESTful 서비스가 이를 제공하는 것은 아닙니다. 또한, 이러한 코드는 일반적으로 클라이언트 측에서 실행되므로 보안과 관련된 사항이 필히 고려되어야 합니다.

지금까지 REST API의 개요를 알아보았습니다. 다음 편에서는 REST API가 어떻게 동작하는지에 관한 구체적 설명을 이어가겠습니다.

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.