PRESENTATION

16주차 주제 REST

REST ~무상태성~

by mun

REST 란?

Representational State Transfer

월드 와이드 웹과 같은 분산 하이퍼미디어 시스템을 위한
소프트웨어 아키텍처의 한 형식, 웹 서비스의 설계 모델

효율적, 안정적이며 확장가능한 분산시스템을 가져올 수 있는
소프트웨어 아키텍처 디자인 제약의 모음

자원을 이름으로 구분하여
해당 자원의 상태를 주고받는 모든 것을 의미

REST의 6가지 조건

Code on demand

서버가 클라이언트가 실행시킬 수 있는
로직을 전송하여 기능을 확장시킬 수 있다.

Stateless(무상태성)

각 요청 간 클라이언트의 콘텍스트가 서버
에 저장되어서는 안 된다

Layered System (계층화)

REST 서버는 다중 계층으로 구성될 수 있
다. 클라이언트는 보통 대상 서버에 직접
연결되었는지, 또는 중간 서버를 통해 연
결되었는지를 알 수 없다.

Cacheable (캐시 가능)

캐싱 기능을 적용할 수 있어야 한다.

Self-descriptiveness (자체 표현 구조)

API를 통해 전송되는 내용은 별도 문서 없이
쉽게 이해할 수 있어야 한다

Client - Server 구조

클라이언트와 서버로 분리되어야하며
서로 의존성이 없어야 한다

REST의 6가지 조건

Code on demand

서버가 클라이언트가 실행시킬 수 있는 로직을 전송하여 기능을 확장시킬 수 있다.

Stateless(무상태성)

각 요청 간 클라이언트의 콘텍스트가 서버에 저장되어서는 안 된다

Layered System (계층화)

REST 서버는 다중 계층으로 구성될 수 있다. 클라이언트는 보통 대상 서버에 직접 연결되었는지, 또는 중간 서버를 통해 연결되었는지를 알 수 없다.

Cacheable (캐시 가능)

캐싱 기능을 적용할 수 있어야 한다.

Self-descriptiveness (자체 표현 구조)

API를 통해 전송되는 내용은 별도 문서 없이 쉽게 이해할 수 있어야 한다

Client - Server 구조

클라이언트와 서버로 분리되어야하며 서로 의존성이 없어야 한다

REST 무상태성

HTTP를 베이스로 한 무상태성(stateless) 클라이언트/서버 프로토콜

시스템이 현재의 상태를 나타내는 데이터를 보유하지 않고,
단지 입력 내용에 따라 출력 내용이 결정되는 방식

REST API는 서버로의 요청(reqeust)이나
클라이언트로의 응답(response)이
모두 무상태성이란 특징을 가져야 한다.

REST 무상태성

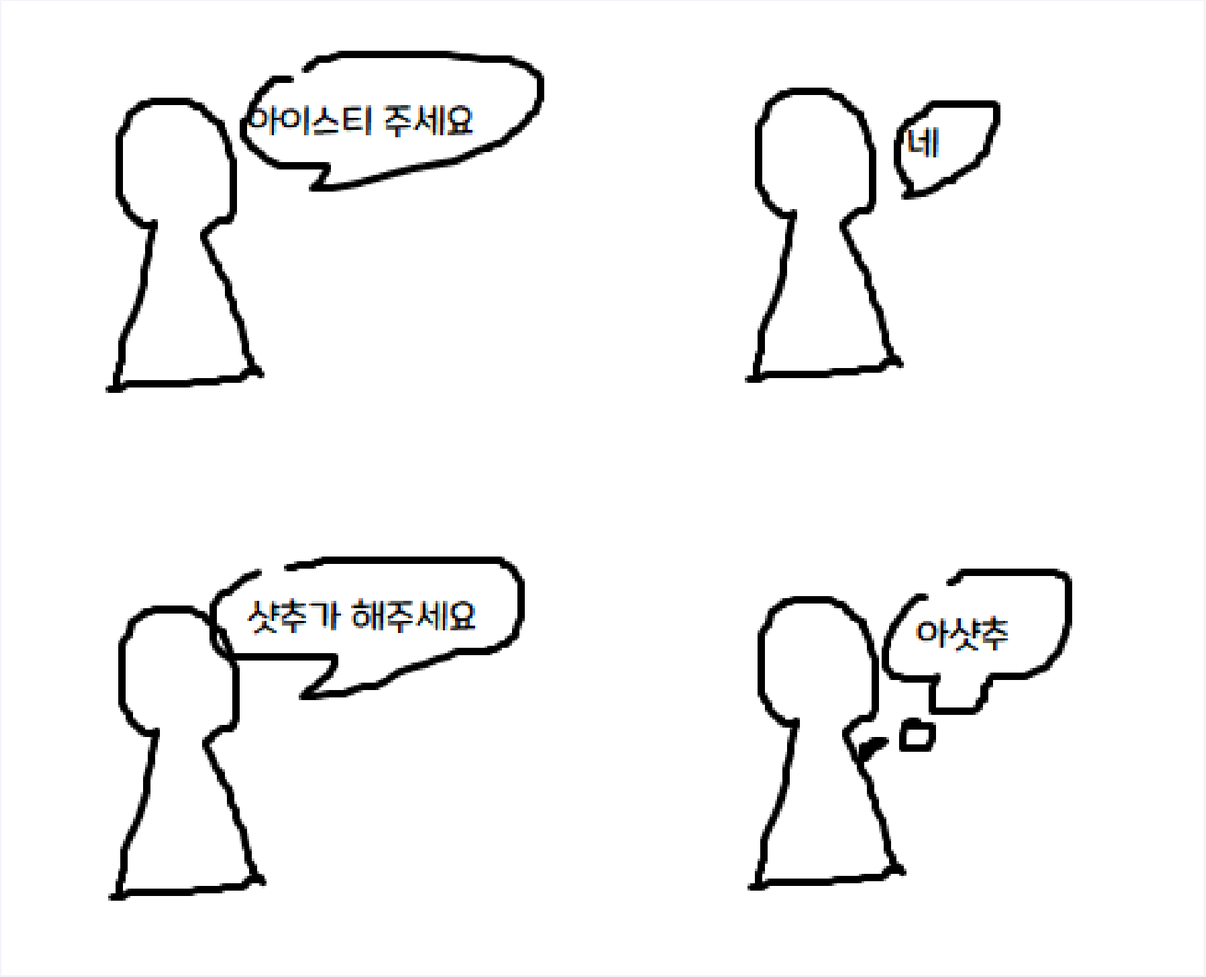
HTTP를 베이스로 한 무상태성(stateless) 클라이언트/서버 프로토콜

시스템이 현재의 상태를 나타내는 데이터를 보유하지 않고,
단지 입력 내용에 따라 출력 내용이 결정되는 방식

REST API는 서버로의 요청(reqeust)이나
클라이언트로의 응답(response)이
모두 무상태성이란 특징을 가

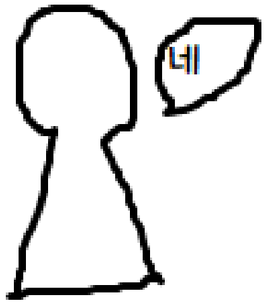
클라이언트의 상태는 서버에 저장되지 않고
각각의 요청은 클라이언트가 필요로 하는
모든 정보를 포함하고 있어야 한다

상태성과 무상태성

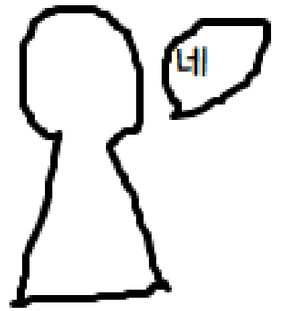


상태성

상태성과 무상태성

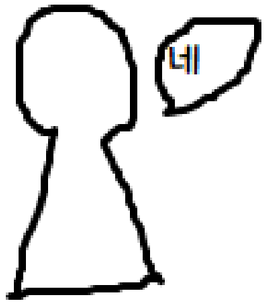


상태성

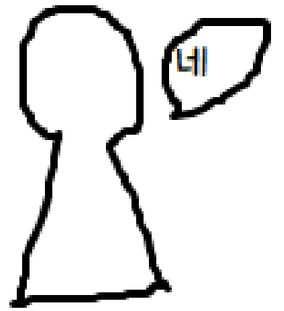


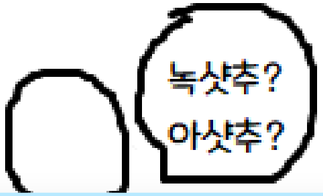
무상태성

상태성과 무상태성



상태성





무상태성

각각의 요청을 완전히 별개의 것으로 인식하고 처리

상태성 VS 무상태성

서버가 클라이언트의 이전 요청을 기억하고 있다

장점

요청이 간결하다

서버가 클라이언트의 요청 내역을 기억한다

상태성 VS 무상태성

서버가 클라이언트의 이전 요청을 기억하고 있다

장점

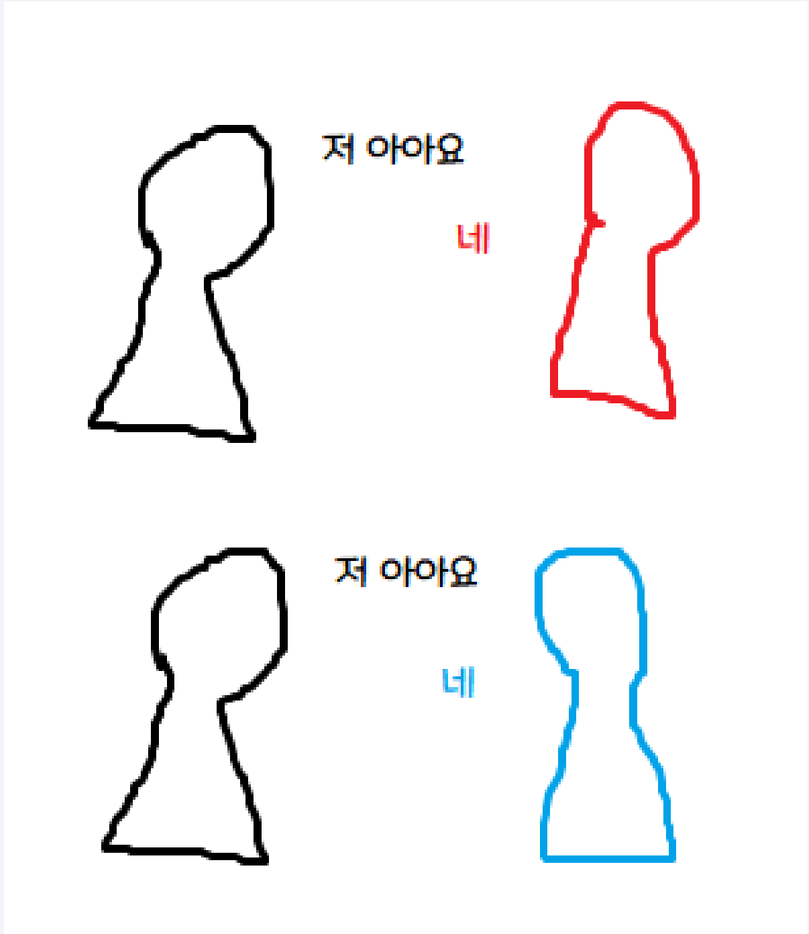
요청이 간결하다
서버가 클라이언트의 요청 내역을 기억한다

단점 확장성

늘어난 요청을 처리하기 위해 서버를 늘리면,
늘린 서버에서 어떤 클라이언트의 요청이든 기억하고
처리할 수 있어야 하기 때문에,
서버 간 애플리케이션 상태가 동기화 되어야 한다

상태성 VS 무상태성

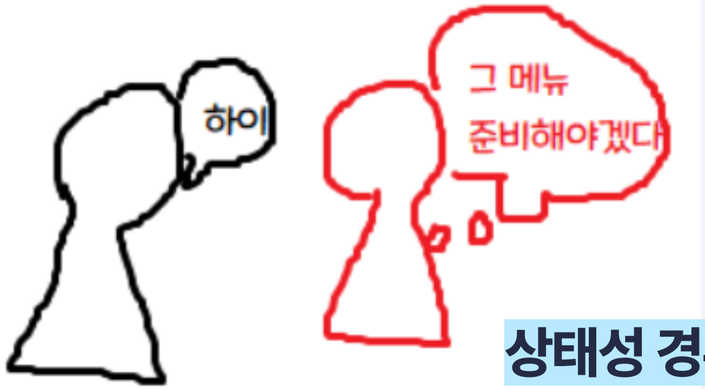
이전 요청과 현재 요청이 관계가 없다



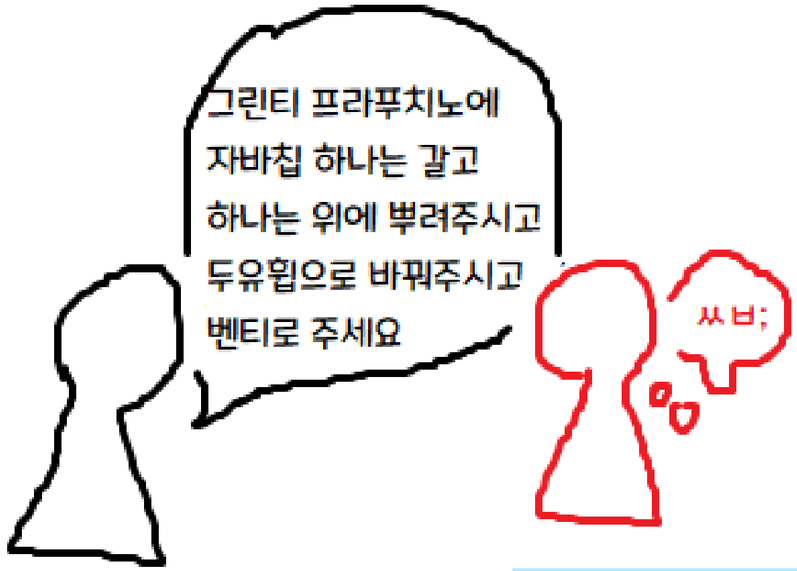
장점 확장성

매 요청마다 필요한 모든 정보를 포함하고 있어
서버는 이전 요청을 기억할 필요가 없기 때문에,
현재 요청만 처리하면 되고,
이로 인해서 서버 시스템이 단순해져 서버 확장하기 용이하다

상태성 VS 무상태성



상태성 경우



무상태성 경우

이전 요청과 현재 요청이 관계가 없다

장점 확장성

매 요청마다 필요한 모든 정보를 포함하고 있어
서버는 이전 요청을 기억할 필요가 없기 때문에,
현재 요청만 처리하면 되고,
이로 인해서 서버 시스템이 단순해져 서버 확장하기 용이하다

단점 퍼포먼스 저하

한번 요청시 Stateful보다 송신할 데이터의 양이 많다
인증이 필요한 경우 서버에 부하가 걸린다

무상태성은 무조건 지켜져야 할까?

세션 = 서버에 저장된다 = REST 위반

BUT... 세션을 사용함으로서 얻는 이점도 있다

무상태성은 무조건 지켜져야 할까?

세션 = 서버에 저장된다 = **REST 위반**

BUT... **세션을 사용함으로서 얻는 이점도 있다**

환경에 따라 일부 원칙을 포기하고
적합한 방법을 사용할 수 있다

REST를 이해하고
좋은 REST API를 개발해보아요

감사합니다

참고 :
https://ko.wikipedia.org/wiki/REST,
https://khj93.tistory.com/entry/%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-REST-API%EB%9E%80-REST-RESTful%EC%9D%B4%EB%9E%80,
https://postitforhooney.tistory.com/entry/RestfulRESTful%EC%9D%B4%EB%9E%80-%ED%8D%BC%EC%98%B4 , https://velog.io/@yhlee9753/REST-API-%EC%99%80-HTTP-Stateless-%EC%97%90-%EB%8C%80%ED%95%9C-%EA%B3%A0%EC%B0%B0