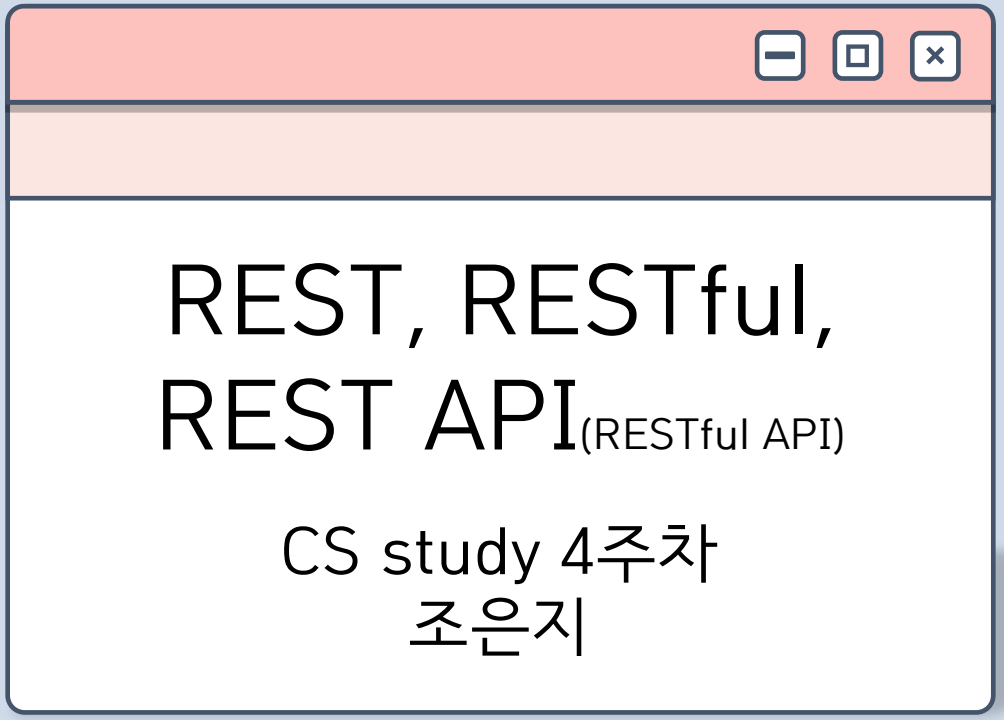
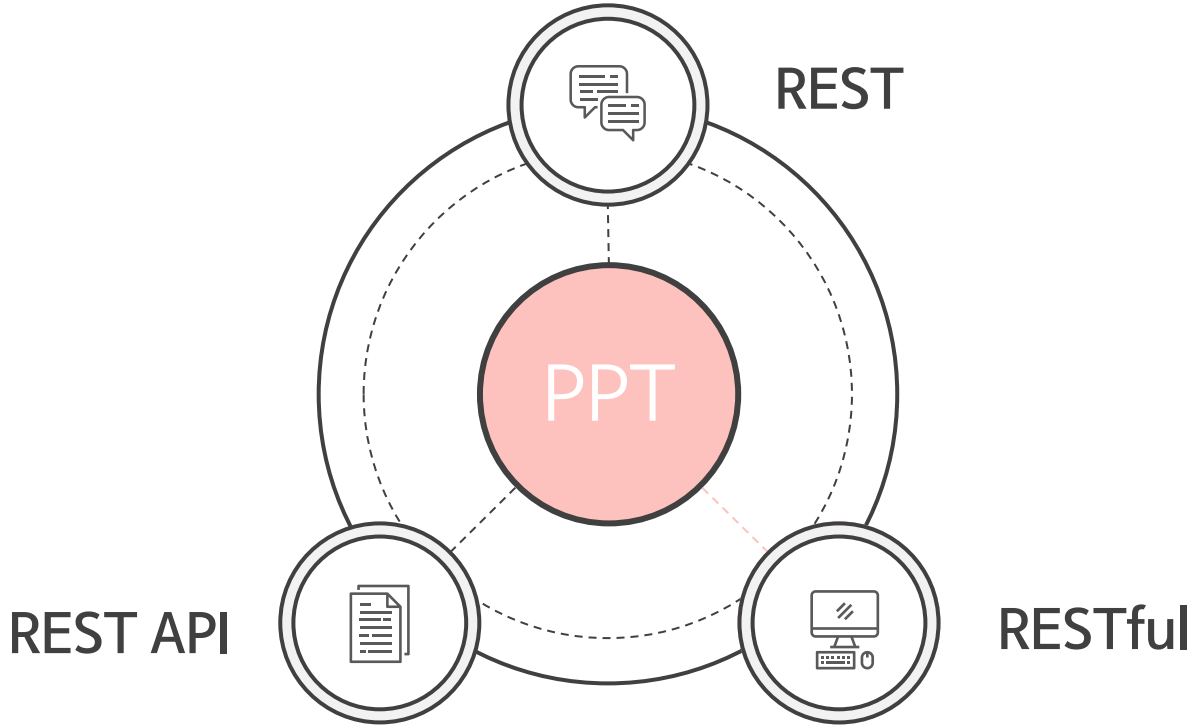


REST, RESTful, REST API(RESTful API)

CS study 4주차
조은지

REST, REST API, RESTful



REST?

REpresentational State Transfer
네트워크 상에서 client와 server 사이의 통신 방식 중 하나
URI와 HTTP를 이용한 통신 목적의 아키텍처 스타일
Application 사이에 결합도를 낮추게끔 설계하는 아키텍처 스타일

아키텍처?
제약 조건을 모두 만족하는 시스템

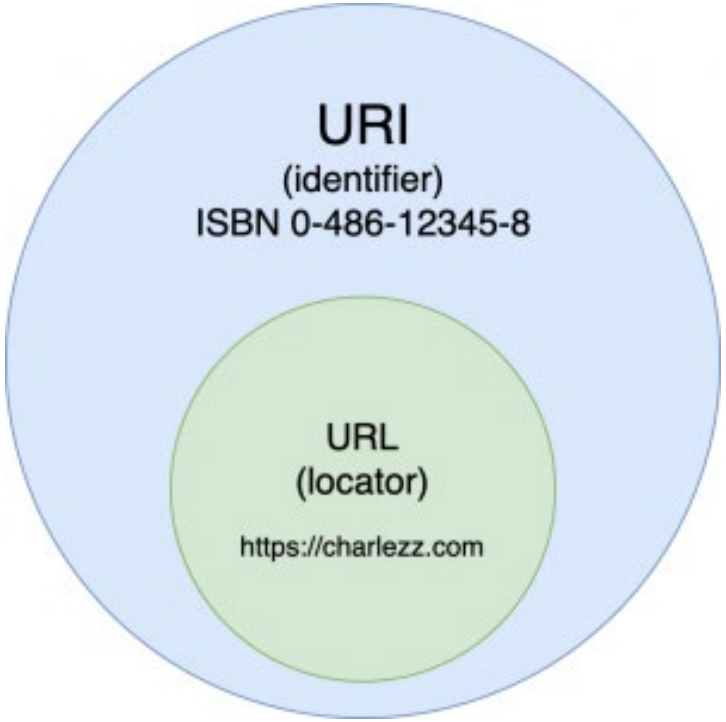
URI

특정 리소스를 식별하는 통합 자원 식별자.

웹 기술에서 사용하는 논리적 또는 물리적 리소스를 식별하는 고유한 문자열 시퀀스

URL

흔히 웹 주소라고 하며, 컴퓨터 네트워크 상에서 리소스가 어디에 있는지 알려주기 위한 규약.



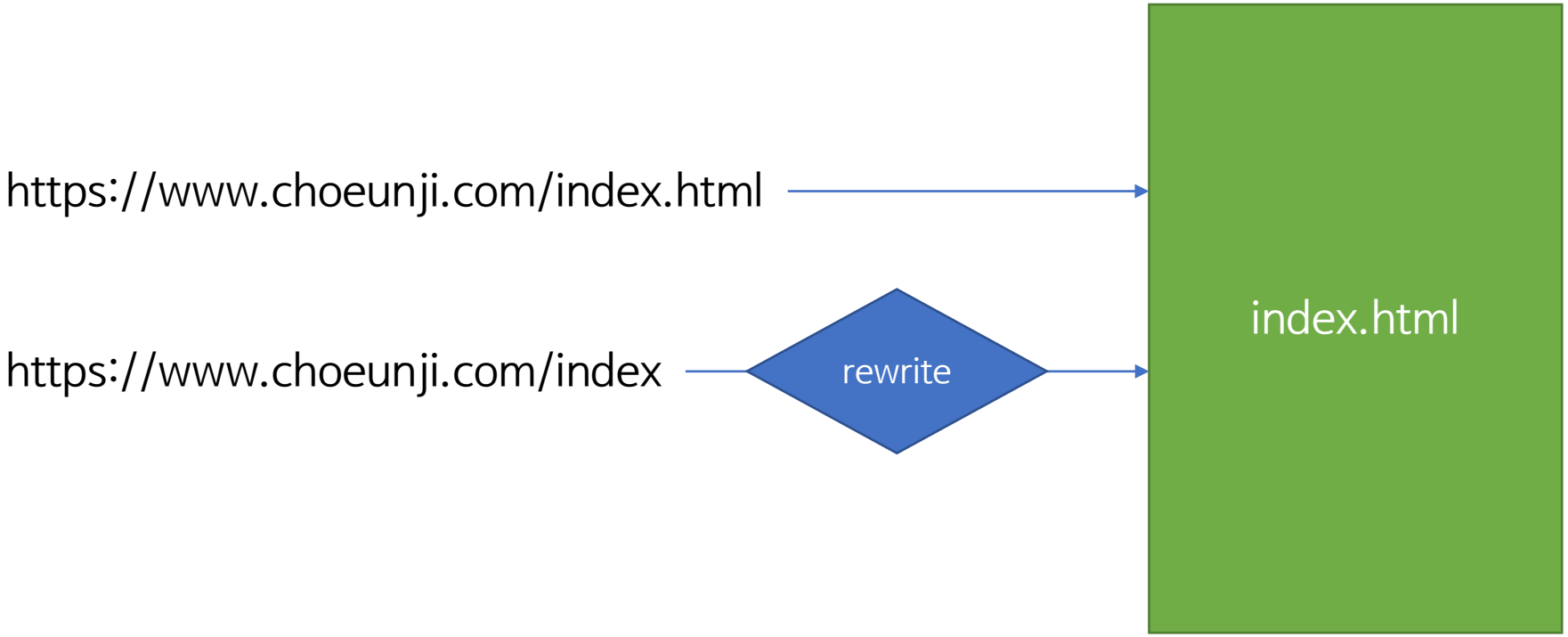
URI vs URL

URI와 URL의 가장 큰 차이점은 **URI는 식별**하고, **URL은 위치**를 가리킨다는 것!

예를 들어 “조은지”는 이름이며 식별자이다. 이것은 나이나 주소 등의 정보가 없으므로 URI는 될 수 있지만, URL은 될 수가 없다.

“경기 성남시 분당구 불정로 6”는 주소이다. 주소는 특정 위치를 가리키고, 식별자이므로 URI와 URL이 될 수 있다.

REST, REST API, RESTful



~index.html은 웹 서버의 실제 파일 위치를 나타내는 주소이므로 URI와 URL 둘 다 될 수 있지만
~index는 index라는 파일이 웹서버에 존재하지 않으므로 URL은 아니다.

REST?

자원을 이름으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것

즉, 자원의 표현을 가지고 상태 전달을 한다~ 를 의미함

구분	정의
자원	해당 소프트웨어가 관리하는 모든 것(문서, 그림, 데이터 등)
표현	그 자원을 표현하기 위한 이름(DB의 '학생 정보'가 자원이라면, 그 이름인 students를 자원의 표현이라고 정함)
상태 전달	데이터가 요청되는 시점에 자원의 상태를 전달함(JSON, XML을 통해 데이터를 주고 받는 것이 일반적이다)

HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고, HTTP Method(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 CRUD를 적용하는 것을 의미한다.

HTTP method

상태 전달을 하기 위해 HTTP 프로토콜의 method를 이용함

분류	설명
GET	정보 요청, URI가 가진 정보를 검색하기 위해 서버에 요청한다. (READ)
POST	정보 입력, client에서 serve로 전달하려는 정보를 보낸다. (CREATE)
PUT	정보 업데이트, 주로 내용을 갱신하기 위해 사용한다. (UPDATE)
DELETE	정보 삭제(DELETE)
HEAD	GET과 동일하지만 서버에서 Body를 return하지 않음. 오직 찾기만을 원할 때 사용. 찾는 Object가 존재할 경우 응답의 상태코드만을 확인한다.
TRACE	Client로 부터 Request Packet이 방화벽, Proxy Server, Gateway등을 거치면서 packet의 변조가 일어날 수 있는데, 이 때 Server에 도달 했을 때의 최종 Packet의 Request Packet을 볼 수 있다. Original 데이터와 서버에 도달했을 때의 데이터를 서버의 응답 Body를 통해 확인할 수 있다.
OPTIONS	Target server의 지원가능한 method를 알아보기 위함
CONNECT	웹 서버에 프록시 기능을 요청할 때 사용된다.

REST의 조건

1.일관된 인터페이스

2.Client-Server

3.Stateless

4.캐시 가능

5.계층화된 시스템

6. Code-on-Demand (선택)

1. 일관된 인터페이스

전체적인 시스템 아키텍처를 간단하고 잘 파악할 수 있도록 하기 위한 약속된 interface를 사용한다.

개발 REST 규약으로 4가지가 있다.

- 1) identification of resources / 자원은 유일하게 식별 가능해야 한다. (URI)
- 2) manipulation of resources through representations / representations(get,put,delete 등) 전송을 통해서 자원을 조작해야 한다.
- 3) self-descriptive messages / 메시지만 보고 무슨 뜻인지 알아야 한다.
- 4) hypermedia as engine of application state / application의 상태는 hyperlink를 이용해서 전이되어야 한다.

4) application의 상태는 hyperlink를 이용해서 전이되어야 한다.

HTML은 링크를 통해 상태 전이가 될 수 있다.
a 태그를 통해서 Hyperlink를 사용한 상태 전이가 가능하다.

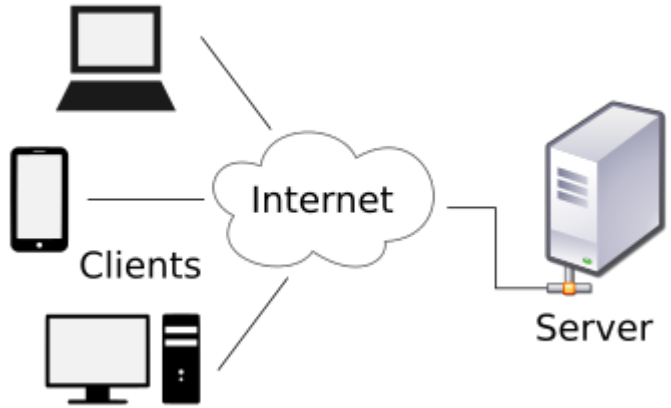
HTTP/1.1 200 OK
Content-Type: text/html
<html>
<head></head>
<body>**test**</body>
<html>

JSON은 HTTP의 Link header를 통해서 상태전이가 가능하다.

HTTP/1.1 200 OK
Content-Type: application/json
Link: </article/1>; rel="previous",
</article/3>; rel="next";
{
"title": "The second article",
"contents": "blah blah.."
}

2. Client - Server

client가 server에게 request를 보내고, server가 client에게 response를 보내는 구조
server는 API 제공과 제공된 API를 이용하여 비즈니스 로직을 처리하거나 저장하는 역할
client는 사용자 인증이나 세션, 로그인 정보 등을 관리하는 역할
→ 각각의 역할을 나누어 의존성을 줄인다.



3. Stateless

HTTP 프로토콜은 Stateless 프로토콜이므로 REST도 마찬가지이다.

모든 요청은 필요한 모든 정보를 담고 있어야 한다.

client의 환경 정보를 server에 저장하지 않는다. (따라서 세션의 정보는 전적으로 client가 가지고 있어야!)

server는 각각의 요청을 완전히 별개의 것으로 인식하고 처리한다.

- 즉, 이전 요청이 다음 요청의 처리에 연관되어서는 안된다.

4. 캐시 가능

HTTP 프로토콜을 그대로 사용하므로 웹에서 사용하는 기존의 인프라를 그대로 사용할 수 있다.

요청에 대한 응답 내의 데이터에 캐시가 가능한지 불가능한지 명시해야 한다.

응답을 캐시할 수 있다면 클라이언트에서 동일한 요청이 왔을 때 응답 데이터를 재사용할 수 있어야 한다.

대량의 요청을 효율적으로 처리할 수 있다.

5. 계층화된 시스템

서버는 API 서버에 여러 계층을 추가하여 유연한 구조로 개발될 수 있다.

client는 REST API server만 호출한다. 하지만, 서버는 다중 계층으로 구성될 수 있다.

ex) 순수 비즈니스 로직을 수행하는 API 서버와 그 앞 단에 사용자 인증, 암호화 등을 하는 계층을 추가해 구조상의 유연성을 둘 수 있다.

6. Code-on-Demand (선택, 권장x)

Server가 리소스에 대한 표현을 어떻게 처리해야 하는지에 대한 code를 제공한다.

Java Applet, JavaScript, Flash 등이 Code On Demand의 예시

-> 서버에서 제공되는 코드를 실행해야 하기 때문에 보안 문제를 야기할 수 있음

REST API(RESTful API)?

REST가 적용된 API
REST의 특징을 기반으로 서비스 API를 구현하는 것
REST의 설계 규칙을 잘 지켜서 설계된 API

각 요청이 어떤 동작이나 정보를 위한 것인지 그 요청의 모습 자체로 추론이 가능하다.

REST API 설계 규칙

1. URI는 명사를 사용한다.
    - /getUserId 등을 사용하지 않는다!
2. 슬래시로 계층 관계를 표현한다.
    - notice/search
3. URI 마지막 문자로 슬래시를 포함하지 않는다.
4. 언더바를 사용하지 않고, 하이픈을 사용한다.
5. URI는 소문자로만 구성한다.
6. HTTP 응답 상태 코드를 사용한다.
    - client는 해당 요청에 대한 실패, 처리 완료 또는 잘못된 요청 등에 대한 피드백을 받아야 한다.
    - 예를 들어 400번대는 client 오류, 500번대는 서버 오류
7. 파일 확장자는 URI에 포함하지 않는다.

REST API를 사용하는 이유

1. 분산 시스템 설계를 위해서

   거대한 어플리케이션을 모듈, 기능별로 분리하기 쉬워짐

   REST API를 사용하면 어떤 다른 모듈 또는 어플리케이션이라도 REST API를 통해 상호간 통신을 할 수 있기 때문!

2. Web 브라우저 외의 클라이언트를 위해서

   웹 페이지를 띄우기 위해 html을 보내던 것과 달리 데이터만 보내면 여러 클라이언트에서 해당 데이터를 적절하게 보여주기만 하면 된다.

   요청한 데이터만 보내주면 되기 때문에 서버가 가벼워지고, 유지보수성이 좋아진다.

RESTful?

REST가 적용된 시스템

REST의 원리를 잘 따르는 시스템

REST와 HTTP는 다르다!

REST는 HTTP에서 제약조건이 추가된 것

개발 REST 규약으로 4가지

1) identification of resources / 자원은 유일하게 식별 가능해야 한다. (URI)

2) manipulation of resources through representations / representations(get,put,delete 등) 전송을 통해서 자원을 조작해야 한다.

3) self-descriptive messages / 메시지만 보고 무슨 뜻인지 알아야 한다.

4) hypermedia as engine of application state / application의 상태는 hyperlink를 이용해서 전이되어야 한다.

실무에서 REST의 원칙을 모두 지키면서 개발하는 것은 현실적으로 힘들고 추가 개발 비용대비 효과가 있는 것이 아니다.

보통 REST의 원칙을 완벽하게 지키지 않아도 REST API라고 한다.