**💡 라우팅 유형에는 어떤 것들이 있나요?**

라우터가 라우팅 테이블을 만드는 방식에 따라 라우팅은 2가지 유형으로 구분됩니다.

**정적 라우팅**

정적 라우팅에서는 네트워크 관리자가 정적 테이블을 사용하여 네트워크 경로를 수동으로 구성하고 선택합니다. 정적 라우팅은 네트워크 설계나 파라미터가 일정하게 유지될 것으로 예상되는 경우에 유용합니다.

이 라우팅 기법에서는 정적 특성으로 인해 네트워크 정체와 같은 단점이 나타날 수 있습니다. 링크에서 장애가 발생할 경우에 대비해 관리자가 대체 경로를 구성할 수도 있지만, 일반적으로 정적 라우팅은 네트워크의 적응성과 유연성을 저하시켜 네트워크 성능을 제한합니다.

**동적 라우팅**

동적 라우팅에서 라우터는 실제 네트워크 조건에 따라 런타임에 라우팅 테이블을 만들고 업데이트합니다. 동적 라우팅 테이블을 만들고 유지 관리하고 업데이트하는 규칙 집합인 동적 라우팅 프로토콜을 사용하여 소스에서 대상까지 가장 빠른 경로를 찾으려고 시도합니다.

동적 라우팅의 가장 큰 장점은 트래픽 볼륨, 대역폭, 네트워크 장애 등 변화하는 네트워크 조건에 대응한다는 것입니다.

**💡 LAN(Local Area Network)**

LAN 은 Local Area Network 의 약자로써 근거리 통신 망을 의미

**💡 네트워크의 종류 및 LAN의 구성형태**

1) 네트워크의 종류

\* PAN ( Personal Area Network ) : 가장 작은 규모의 네트워크

\* LAN ( Local Area Network ) : 근거리 영역 네트워크

\* MAN (Metropolitan Area Network) : 대도시 영역 네트워크

\* WAN (Wide Ares Network) : 광대역 네트워크

\* VAN (Value Added Network) : 부가가치 통신망 정보의 축적과 제공, 통신속도와 형식의 변화, 통신경로의 선택 등 여러 종류의 정보서비스가 부가된 통신망.

\* ISDN (Integrated Services Digital Network) : 종합정보 통신망(=BISDN) 전화, 팩스, 데이터 통신, 비디오텍스 등 통신관련 서비스를 종합하여 다루는 통합서비스 디지털 통신망. 디지털 전송방식과 광섬유 케이블 사용. 꿈의 통신망이라 불립니다.

**LAN ( Local Area Network ) : 근거리 영역 네트워크**

* LAN 은 Local Area Network 의 약자로써 근거리 통신 망을 의미
* "다수의 독립된 컴퓨터 기기들이 상호간에 통신이 가능

**WAN (Wide Ares Network) : 광대역 네트워크**

* 광역 네트워크(WAN)는 사무실, 데이터 센터, 클라우드 애플리케이션 및 클라우드 스토리지를 서로 연결하는 기술
* 하나의 건물이나 대규모 캠퍼스를 넘어 특정 지역이나 심지어 전 세계에 분산된 여러 위치까지도 포함하기 때문에 광역 네트워크라고 합니다

**MAN (Metropolitan Area Network) : 대도시 영역 네트워크**

* MAN 은 LAN 보다는 큰 규모를 가지지만 WAN 보다는 지리적으로 작은 규모에서 컴퓨터 자원들과 HOST들을 연결하는 도시권 통신망이라 합니다.

**VAN (Value Added Network) :**

* 부가가치 통신망 정보의 축적과 제공, 통신속도와 형식의 변화, 통신경로의 선택 등 여러 종류의 정보서비스가 부가된 통신망.

**네트워크의 공간 크기 개념 : WAN > MAN > LAN > PAN**

**💡 허브(HUB)란?**

컴퓨터와 컴퓨터 사이, 즉 네트워크 장비와 장비를 연결해 주는 기능을 수행하는 장비입니다.

허브는 일종의 멀티포트 리피터(Multiport Repeater)의 기능을 수행하기 때문에 서로 다른 네트워크 장비를 연결해 줄 수 있다.

-멀티 포트 : 다양한 기기로부터 오는 케이블을 연결 할 수 있는 인터페이스를 제공한다는 의미

- 리피터 : 서로 다른 네트워크 기기에서 오는 신호들을 증폭시켜서 전달해준다는 의미

즉, 허브는 다양한 기기들로부터 오는 전기신호들을 받아서 그 신호들을 증폭시켜 다른 기기들로 뿌려주는 역할.

**💡 패킷(packet)의 정의**

패킷이란 네트워크에서 출발지와 목적지간에 라우팅되는 데이터의 단위.

즉, 정보를 보낼 때 특정 형태를 맞추어 보내 컴퓨터 간에 데이터를 주고 받을 때 네트워크를 통해 전송되는 데이터 조각

**💡 [Web] 웹 서버(Web Server)와 WAS의 차이**

**웹 서버(Web Server)**

* 웹 서버란 HTTP 프로토콜을 기반으로 클라이언트가 웹 브라우저에서 어떠한 요청을 하면 그 요청을 받아 정적 컨텐츠를 제공하는 서버이다. 정적 컨텐츠란 단순 HTML 문서, CSS, 이미지, 파일 등 즉시 응답 가능한 컨텐츠이다.
* 이때 웹 서버가 정적 컨텐츠가 아닌 동적 컨텐츠를 요청받으면 WAS에게 해당 요청을 넘겨주고, WAS에서 처리한 결과를 클라이언트에게 전달하는 역할도 해준다.
* 이러한 웹 서버에는 Apache, NginX 등이 있다.

**WAS(Web Application Server)**

* WAS란 DB 조회 혹은 다양한 로직 처리를 요구하는 동적 컨텐츠를 제공하기 위해 만들어진 Application 서버이다. HTTP 프로토콜을 기반으로 사용자 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어로서, 주로 데이터베이스 서버와 같이 수행된다.
* WAS는 JSP, Servlet 구동환경을 제공해주기 때문에 서블릿 컨테이너 혹은 웹 컨테이너로 불린다.
* 이러한 WAS는 웹 서버의 기능들을 구조적으로 분리하여 처리하고자 하는 목적으로 제시되었다.
* 분산 트랜잭션, 보안, 메시징, 쓰레드 처리 등의 기능을 처리하는 분산 환경에서 사용된다.
* WAS는 프로그램 실행 환경과 DB 접속 기능을 제공하고, 여러 개의 트랜잭션을 관리 가능하다. 또한 비즈니스 로직을 수행할 수 있다.
* 이러한 WAS에는 Tomcat, JBoss, WebSphere 등이 있다.

웹 서버와 WAS를 하는 이유

**1. 서버 부하 방지**

WAS와 웹 서버는 분리하여 서버의 부하를 방지해야 한다. WAS는 DB 조회나 다양한 로직을 처리하고, 단순한 정적 컨텐츠는 웹 서버에서 처리해줘야 한다. 만약 정적 컨텐츠까지 WAS가 처리한다면 부하가 커지게 되고, 수행 속도가 느려질 것이다.

**2. 보안 강화**

SSL에 대한 암호화, 복호화 처리에 웹 서버를 사용 가능

**3. 여러 대의 WAS 연결 가능**

로드 밸런싱을 위해 웹 서버를 사용할 수 있다. 여러 개의 서버를 사용하는 대용량 웹 어플리케이션의 경우 웹 서버와 WAS를 분리하여 무중단 운영을 위한 장애 극복에 쉽게 대응할 수 있다.

**4. 여러 웹 어플리케이션 서비스 가능**

하나의 서버에서 PHP, JAVA 애플리케이션을 함께 사용할 수 있다.

이러한 이유로 웹 서버를 WAS 앞에 두고 필요한 WAS들을 웹 서버에 플러그인 형태로 설정하면 효율적인 분산 처리가 가능하다.

**💡 캐시/ 웹 캐시(Cache)란?**

캐시(Cache)란 자주 사용하는 데이터나 값을 미리 복사해 놓는 임시 장소를 말한다.

캐시는 캐시의 접근 시간에 비해 원래 데이터를 접근하는 시간이 오래 걸리는 경우나 값을 다시 계산하는 시간을 절약하고 싶은 경우에 사용한다.

캐시에 데이터를 미리 복사해 놓으면 계산이나 접근 시간 없이 더 빠른 속도로 데이터에 접근할 수 있다.

**지역성(Locality)**

모든 데이터를 캐시에 담기에는 저장 공간이 그리 크지 않기 때문에 힘들다. 그래서 보통 캐시는 지역성을 나누어 분류한다.

**시간적 지역성**

특정 데이터가 한번 접근되었을 경우 가까운 미래에 또 한번 데이터에 접근할 가능성이 높은 것을 말한다. 즉, 한 번 가져왔던 데이터를 또 쓸 일이 있다는 의미를 말한다.

이런 경우 캐시에 한번 가져와서 저장 해놓고 여러 번 사용하게 되면 메모리에 접근하는 횟수가 줄어든다. 따라서 캐시는 반복적으로 사용되는 데이터가 많을수록 높은 효율성을 낼 수 있다.

**공간적 지역성**

특정 데이터와 가까운 주소가 순서대로 접근되었을 경우 공간적 지역성이라고 한다. CPU 캐시나 디스크 캐시의 경우에는 한 메모리 주소에 접근할 때 그 주소뿐 아니라 해당 블록을 전부 캐시에 가져오게 된다.

이 때 메모리 주소를 오름차순이나 내림차순으로 접근한다면 캐시에 이미 저장된 같은 블록의 데이터를 접근하게 되므로 캐시의 효율성이 크게 향상된다.

다시 말해 앞으로 사용할 데이터들이 가져올 블록안에 모여있는 것을 말하는 것이다. 필요한 데이터가 모여 있다면 한번만 메모리에 접근해도 필요한 데이터들을 가져올 수 있다는 것이다.

만약 데이터가 모여 있지 않다면 Cache Miss가 날 확률이 높아지고 메모리에 여러 번 접근하게 되어 효울성이 떨어지게 된다.

**💡 웹 캐시(Web Cache)란?**

웹 캐시는 사용자(Client)가 웹 사이트(Serve)에 접속할 때, 정적 컨텐츠(이미지, JS, CSS 등)를 특정 위치(Client, Network 등)에 저장하여,

웹 사이트 서버에 해당 컨텐츠를 매번 요청하여 받는 것이 아니라, 특정 위치에서 불러옴으로써 사이트 응답시간을 줄이고 서버 트래픽 감소 효과를 볼 수 있는 것을 말한다.

**💡 URI, URL, URN이란?**

텍스트, 폰트, 스크린샷, 브랜드이(가) 표시된 사진

자동 생성된 설명

**URI(Uniform Resource Identifier)란?**

자원의 식별(Identifier)을 말한다

통합 자원 식별자를 말한다.

인터넷에 있는 자원을 나타내는 유일한 주소를 말한다.

URI의 존재는 인터넷에서 요구되는 기본조건으로 인터넷 프로토콜에 항상 붙어다닌다.

하위 개념으로는 URL, URN이 있다. (즉 URI는 상위개념)

**URL(Uniform Resource Locator)이란?**

자원의 위치를 말한다.

프로토콜(스키마)을 포함한 개념이다.

네트워크 상에서 자원이 어디있는지(Path)를 알려주기 위한 규약을 말한다.

일반적으로 사이트 도메인을 자주 의미한다.

웹 상 뿐만 아니라 컴퓨터 네트워크 상의 자원은 모두 나타낼 수 있다.

**URN(Uniform Resource Name)이란?**

자원의 이름을 말한다.

프로토콜(스키마)을 포함하지 않은 개념이다.

해당 자원의 이름(Name)을 의미한다.

자원에 대하여 영속적이고 유일하다.

위치에 독립적인 이름을 제공하기 위하여 존재한다.

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

**💡 [HTTP 메서드 종류]**

1. GET
2. POST
3. PUT
4. PATCH
5. DELETE

**💡 DNS, DDNS란 무엇일까?**

**DNS(Domain Name System)란?**

DNS는 도메인 이름 시스템을 말한다.

DNS는 IP라는 까다로운 체계로 되어있는 주소를 사용자가 보다 쉽게 이용할 수 있도록 도메인으로 변경해주는 역할을 하는 시스템이라고 보면 될 것 같다.

**DDNS(Dynamic Domain Name System)란?**

동적 DNS를 의미한다.

유동IP 환경에서 DNS로 도메인을 설정하였을 때 계속 변경되는 IP를 자동으로 설정 되어있는 자동으로 DNS서버의 IP를 갱신해주는 기능을 가진 시스템이라고 보면 될 것 같다.

**💡 REST란?**

REST(Representational State Transfer)의 약자로 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 모든 것을 의미합니다.

1. HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고,
2. HTTP Method(POST, GET, PUT, DELETE, PATCH 등)를 통해
3. 해당 자원(URI)에 대한 CRUD Operation을 적용하는 것을 의미합니다.

**REST의 특징**

1. Server-Client(서버-클라이언트 구조)
2. Stateless(무상태)
3. Cacheable(캐시 처리 가능)
4. Layered System(계층화)
5. Uniform Interface(인터페이스 일관성)

**REST의 장단점**

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 해 준다.
* 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.
* 여러 가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.
* 서버와 클라이언트의 역할을 명확하게 분리한다.

단점

* 표준이 자체가 존재하지 않아 정의가 필요하다.
* HTTP Method 형태가 제한적이다.
* 브라우저를 통해 테스트할 일이 많은 서비스라면 쉽게 고칠 수 있는 URL보다 Header 정보의 값을 처리해야 하므로 전문성이 요구된다.
* 구형 브라우저에서 호환이 되지 않아 지원해주지 못하는 동작이 많다.(익스폴로어)

**💡 REST API란?**

RESPT API란 REST의 원리를 따르는 API를 의미합니다.

**REST API 설계 예시**

1. URI는 동사보다는 명사를, 대문자보다는 소문자를 사용하여야 한다.

2. 마지막에 슬래시 (/)를 포함하지 않는다.

3. 언더바 대신 하이폰을 사용한다.

4. 파일확장자는 URI에 포함하지 않는다.

5. 행위를 포함하지 않는다.

**장점 (Positive Effect)**

1. Easy to use (쉬운 사용)

REST API 메시지를 읽는 것 만으로도 메시지가 의도하는 바를 명확하게 파악

해당 URI와 원하는 메소드 자체만 독립적으로 이해

2. Complete Seperation between Client and Server\

각자의 역할이 명확하게 분리

HTTP 프로토콜 서비스라는 기본적인 요구만 충족되면 다양한 플랫폼에서 원하는 서비스를 쉽고 빠르게 개발하고 배포할 수 있게 됩니다.

3. Detail expression for specific data type

특정 메소드의 세부적인 표현 문구를 JSON, XML 등 다양한 언어를 이용하여 작성할 수 있다는 장점 뿐만 아니라, 간결한 헤더 표현을 통한 가독성 향상이라는 두마리 토끼를 잡는 효과

**단점 (Negative Effect)**

1. Restriction of HTTP Method

메소드 형태가 제한적

2. Absence of Standard (표준의 부재)

관리의 어려움과 좋은(공식화 된) API 디자인 가이드가 존재하지 않음

**💡 RESTful이란?**

RESTFUL이란 REST의 원리를 따르는 시스템을 의미합니다. 하지만 REST를 사용했다 하여 모두가 RESTful 한 것은 아닙니다. REST API의 설계 규칙을 올바르게 지킨 시스템을 RESTful하다 말할 수 있으며

모든 CRUD 기능을 POST로 처리 하는 API 혹은 URI 규칙을 올바르게 지키지 않은 API는 REST API의 설계 규칙을 올바르게 지키지 못한 시스템은 REST API를 사용하였지만 RESTful 하지 못한 시스템이라고 할 수 있습니다.

**💡 HTTP 응답코드**

**<200 번대 응답(Response) : 성공(Success)>**

1. 200 OK

\* 요청 정상 처리.

1. 204 No Content

\* 요청 정상 처리하였지만, 돌려줄 리소스 없음.

\* 응답에 어떠한 엔티티 바디(Entity Body)도 포함하지 않음.

\* 서버에서 처리 후, 클라이언트에 정보를 보낼 필요가 없는 경우 사용.

1. 206 Partial Content

\* Range가 지정된 요청인 경우, 지정된 범위만큼의 요청을 받았다는 것을 알려줌.

**<300 번대 응답(Response) : 리디렉션(Redirection)>**

1. 301 Moved Permanently

\* 요청된 리소스에는 새로운 URI가 지정되어 있기 때문에, 이후로는 새 URI를 사용해야 한다는 것을 나타냄. (영구적인 URI 변경)

1. 302 Found

\* 요청된 리소스에는 새로운 URI가 지정되어 있기 때문에, 이후로는 새 URI를 사용해야 한 다는 것을 나타냄. 301과 유사하지만, 302는 일시적인 URI 이동)

1. 303 See Other

\* 이 응답은 요청에 대한 리소스는 다른 URI에 있기 때문에 GET 메서드를 사용해서 얻어야 한다는 것을 나타냄. 302 코드와 같지만, 303은 리디렉션 위치를 GET 메서드를 통해 얻어야 한다고 명확하게 되어 있음.

1. 304 Not Modified

\* 요청한 리소스가 마지막 요청 이후 변경된 적이 없기 때문에 기존 클라이언트의 로컬 캐시 리소스를 사용하도록 알려줌.

300번대로 분류되어 있지만, 리디렉션과는 관계없는 처리를 함.

1. 307 Temporary Redirect

\* 임시로 페이지를 리다이렉트 함.

**< 400 번대 응답(Response) : 클라이언트 에러 (Client Error) >**

1. 400 Bad Request

\* 클라이언트의 요청 구문이 잘못됨.

\* 브라우저는 이 응답을 200 OK 응답과 동일한 형태로 취급함.

1. 401 Unauthorized

\* 요청 처리를 위해 HTTP 인증(BASIC 인증, DIGEST 인증) 정보가 필요함을 알려줌.

\* 접근 허용을 차단함. 최초 요청에는 인증 다이얼로그 표시하고, 두번째는 인증 실패 응답을 보냄.

1. 403 Forbidden

\* 접근 금지 응답. Directory Listing 요청(서버 파일 디렉토리 목록 표시) 및 관리자 페이지 접근 등을 차단하는 경우의 응답. (파일 시스템 퍼미션 거부, 허가 되지 않은 IP 주소를 통한 액세스의 거부 등)

\* 서버는 엔티티 바디에 접근 거부에 대한 이유를 명시하여 보낼 수 있음.

1. 404 Not Found

\* 클라이언트가 요청한 리소스가 서버에 없음

1. 405 Mothod Not Allowed

\* 허용되지 않는 HTTP 메서드를 사용함.

**< 500 번대 응답(Response) : 서버 에러 (Server Error) >**

1. 500 Internal Server Error

\* 서버에서 클라이언트 요청을 처리 중에 에러가 발생함.

1. 503 Service Unavailable

\* 서버가 일시적으로 요청을 처리할 수 없음.

\* 서버가 과부하 상태이거나 점검중이므로 요청을 처리할 수 없음을 알려줌.

1. 504 Gateway Timeout

\* 서버를 통하는 게이트웨이에 문제가 발생하여 시간이 초과됨.

1. 505 HTTP Version Not Supported

\* 해당 HTTP 버전에서는 지원되지 않는 요청임을 알려줌.

**💡 HTTP 메소드 중 GET방식과 POST방식 차이**

**GET 방식**

GET은 요청을 전송할 때 필요한 데이터를 Body에 담지 않고, 쿼리스트링을 통해 전송합니다.

GET은 불필요한 요청을 제한하기 위해 요청이 캐시될 수 있습니다.

정적 컨텐츠를 요청하고 나면 브라우저에서는 요청을 캐시해두고, 동일한 요청이 발생할 때 서버로 요청을 보내지 않고 캐시된 데이터를 사용합니다.

* GET 요청은 캐시가 가능하다.
* : GET을 통해 서버에 리소스를 요청할 때 웹 캐시가 요청을 가로채 서버로부터 리소스를 다시 다운로드하는 대신 리소스의 복사본을 반환한다. HTTP 헤더에서 cache-control 헤더를 통해 캐시 옵션을 지정할 수 있다.
* GET 요청은 브라우저 히스토리에 남는다.
* GET 요청은 길이 제한이 있다.
* GET 요청은 중요한 정보를 다루면 안된다. (보안)

**‘POST 방식**

POST는 리소스를 생성/변경하기 위해 설계되었기 때문에 GET과 달리 전송해야될 데이터를 HTTP 메세지의 Body에 담아서 전송합니다.

HTTP 메세지의 Body는 길이의 제한없이 데이터를 전송할 수 있습니다. 그래서 POST 요청은 GET과 달리 대용량 데이터를 전송할 수 있습니다.

POST는 데이터가 Body로 전송되고 내용이 눈에 보이지 않아 GET보다 보안적인 면에서 안전하다고 생각할 수 있지만, POST 요청도 크롬 개발자 도구, Fiddler와 같은 툴로 요청 내용을 확인할 수 있기 때문에 민감한 데이터의 경우에는 반드시 암호화해 전송해야 합니다.

그리고 POST로 요청을 보낼 때는 요청 헤더의 Content-Type에 요청 데이터의 타입을 표시해야 합니다.

* POST 요청은 캐시되지 않는다.
* POST 요청은 브라우저 히스토리에 남지 않는다.
* POST 요청은 데이터 길이에 제한이 없다.

**GET 과 POST 의 차이점**

GET과 POST의 특징만 보아도 차이가 나긴하지만 추가적으로 차이점을 정리해보면 다음과 같다.

* 사용목적 : GET은 서버의 리소스에서 데이터를 요청할 때, POST는 서버의 리소스를 새로 생성하거나 업데이트할 때 사용한다.
* DB로 따지면 GET은 SELECT 에 가깝고, POST는 Create 에 가깝다고 보면 된다.
* 요청에 body 유무 : GET 은 URL 파라미터에 요청하는 데이터를 담아 보내기 때문에 HTTP 메시지에 body가 없다. POST 는 body 에 데이터를 담아 보내기 때문에 당연히 HTTP 메시지에 body가 존재한다.
* 멱등성 (idempotent) : GET 요청은 멱등이며, POST는 멱등이 아니다.

**💡 멱등이란?**

멱등의 사전적 정의는 연산을 여러 번 적용하더라도 결과가 달라지지 않는 성질을 의미한다.

GET은 리소스를 조회한다는 점에서 여러 번 요청하더라도 응답이 똑같을 것 이다. 반대로 POST는 리소스를 새로 생성하거나 업데이트할 때 사용되기 때문에 멱등이 아니라고 볼 수 있다. (POST 요청이 발생하면 서버가 변경될 수 있다.)

**💡 JSON과 XML의 차이점은 무엇인가요?**

JSON과 XML은 애플리케이션 간 데이터 교환에 사용되는 데이터 표현입니다.

JSON은 사람과 기계 모두가 읽을 수 있는 오픈 데이터 교환 형식입니다. JSON은 모든 프로그래밍 언어에 독립적이며 매우 다양한 애플리케이션의 일반적인 API 출력입니다.

XML은 데이터를 정의하는 규칙을 제공하는 마크업 언어로, 태그를 사용하여 데이터 속성과 실제 데이터를 구분합니다. 두 형식 모두 데이터 교환에 사용되지만 JSON은 더 새롭고 유연하며 널리 사용되는 옵션입니다.

**💡 JSON과 XML의 유사점은 무엇인가요?**

JSON과 XML은 모두 데이터 직렬화 형식입니다. 이들 형식을 사용하면 표준화된 방식으로 다양한 애플리케이션, 플랫폼 또는 시스템 간에 데이터를 교환할 수 있습니다.

프로그래밍 언어와 플랫폼마다 동일한 데이터를 다르게 표현합니다. 예를 들어 Java 애플리케이션은 데이터 객체를 사용하는 반면, Python 애플리케이션은 동일한 실제 엔터티에 대한 정보를 딕셔너리를 사용하여 저장합니다. 이들 간에 데이터를 교환하려면 다음 단계를 따릅니다.

1. Java 객체를 XML 또는 JSON으로 변환
2. 네트워크를 통해 XML 또는 JSON 파일 전송
3. XML 또는 JSON을 Python 딕셔너리로 다시 변환

변환 데이터는 언어 라이브러리에 내장되어 있으며, 이를 통해 프로그래머는 API를 사용하여 서로 통신하는 애플리케이션을 작성할 수 있습니다. 또한 두 형식 모두 자체 기술 형식이므로, 모든 텍스트 편집기에서 JSON 및 XML 파일을 읽고 편집할 수 있습니다.

XML이 더 오래된 기술이긴 하지만 JSON과 XML 모두 여전히 많이 사용됩니다.

**- 구문**

JSON에 사용되는 구문은 더 간결하며, 쓰고 읽기가 더 쉽습니다. 따라서 객체를 쉽게 정의할 수 있습니다.

**- 구문 분석**

SON은 보다 쉽게 액세스할 수 있는 표준 JavaScript 함수로 구문 분석할 수 있습니다. 게다가 구문과 파일 크기의 차이로 인해 JSON은 XML보다 더 빠르게 구문 분석할 수 있습니다.

**- 보안**

JSON 구문 분석은 XML보다 안전합니다.

XML 구조는 무단 수정에 취약하며, 이로 인해 XML 외부 엔터티 삽입(XXE)이라는 보안 위험이 발생합니다. 또한 비정형 외부 문서 형식 선언(DTD)에도 취약합니다. 전송 시 DTD 기능을 비활성화하면 이 두 가지 문제를 모두 방지할 수 있습니다.

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명