SOAP

🔄 SOAP vs REST API

📌 SOAP API란?

SOAP(Simple Object Access Protocol)은 REST API와 다른 접근 방식을 사용하는 웹 서비스 프로토콜이다.

🆚 주요 차이점 비교

🎯 요청 방식

구분	REST API	SOAP API
요청 구분	HTTP Method + URI	지정된 단일 URI
예시	`GET /books?page=1&size=10`	모든 요청을 하나의 엔드포인트로 전송

📊 데이터 중심 vs 기능 중심

🗂️ REST API (Data Driven)

- **중점**: 어떤 자원(Resource)인가?
- **요청 예시**: `/books?page=1&size=10`
- **응답 형식**: 주로 JSON
- **특징**: 자원 중심의 설계

🛠️ SOAP API (Function Driven)

- **중점**: 어떤 서비스에 요청하는가?
- **요청 형식**: XML
- **HTTP Method**: 주로 POST 사용
- **특징**: 기능 중심의 설계

📋 SOAP 메시지 구조

```
<soap:Envelope>
  <soap:Header>
    <!-- 인증, 트랜잭션, 기타 메타데이터 -->
  </soap:Header>
  <soap:Body>
    <!-- 실제 메시지 본문 -->
  </soap:Body>
</soap:Envelope>
```

🔍 구성 요소

- **Header**: 인증, 트랜잭션, 기타 정보
- **Body**: 메시지 본문
- **메시지 표현**: 어떤 서비스에 요청하는지 명시

📖 WSDL (Web Service Description Language)

🎯 목적

- **대상**: 사람보다는 프로그램이 읽기 위해 작성
- **역할**: 서비스에 대한 상세한 표준화 문서

✅ 장점

- 서비스에 대한 **상세한 표준화** 제공
- 문서 기반으로 클라이언트/서버를 **자동화하여 개발 과정 간소화**

❌ 단점

- **사람이 읽기 어려움**
- WSDL 수정 시 **클라이언트/서버 모두 코드 수정 필요**
- **캐싱이 어려움**

🔒 SOAP이 사용되는 곳

1️⃣ 보안이 매우 중요한 경우

🛡️ 강력한 보안 기능

- **WS-Security** 같은 보안 프로토콜 지원
- 메시지의 **무결성과 기밀성** 보장
- **인증** 시스템 내장

💼 트랜잭션 처리

보안이 중요한 환경에서는 여러 작업이 하나의 트랜잭션으로 처리되어야 한다.

> 🔄 **트랜잭션이란?**
> 여러 작업 실행 시 모두 성공적으로 이루어지거나 아무것도 이루어지지 않도록 하는 원칙

2️⃣ 상태 유지가 필요한 경우

📝 상태 기반 처리

- 트랜잭션 작업에서 **이전 상태를 기억**해야 하는 경우
- 복잡한 비즈니스 로직에서 **상태 정보 유지**가 중요한 경우

📊 종합 비교표

특징	REST API	SOAP API
설계 철학	Data Driven	Function Driven
데이터 형식	JSON (주로)	XML
HTTP Method	GET, POST, PUT, DELETE 등	POST (주로)
캐싱	쉬움	어려움
보안	기본적	강력함 (WS-Security)
상태 관리	Stateless	Stateful 가능
학습 곡선	쉬움	어려움
문서화	사람이 읽기 쉬움	기계가 읽기 쉬움 (WSDL)

🎯 선택 기준

🚀 REST API를 선택해야 하는 경우

- 빠른 개발이 필요한 경우
- 웹 브라우저와의 호환성이 중요한 경우
- 캐싱을 활용하고 싶은 경우
- 간단한 CRUD 작업이 주된 경우

🔐 SOAP API를 선택해야 하는 경우

- 엔터프라이즈급 보안이 필요한 경우
- 복잡한 트랜잭션 처리가 필요한 경우
- 상태 정보 유지가 중요한 경우

- 은행, 금융 등 높은 신뢰성이 요구되는 분야
  REST API