

API

API 기초 완벽 가이드

"만약 너가 이렇게 말하면 이렇게 해줄게!!"

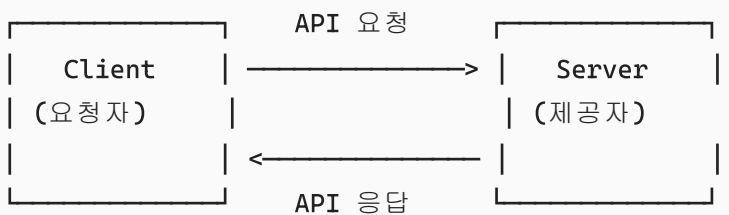
Application Programming Interface (API)란?

API는 소프트웨어들이 서로 대화하는 데 사용하는 통신 규약이다. 마치 서버 역할을 하는 프로그램이 제공하는 나뉜 메뉴판 같은 개념이다.

핵심 개념

- 소프트웨어 간의 통신 담당
- 언어/프레임워크에 제약받지 않음
- 다양한 플랫폼 간 데이터 교환 가능



기본 모델





동작 과정

1. 클라이언트가 API를 통해 서버에 요청
2. 서버가 요청을 처리하고 응답
3. 클라이언트가 응답받은 데이터 활용

API의 주요 용도

기능	설명	HTTP Method
 조회	특정 데이터 가져오기	GET
 생성	새로운 데이터 만들기	POST

기능	설명	HTTP Method
 수정	기존 데이터 변경하기	PUT/PATCH
 삭제	데이터 제거하기	DELETE

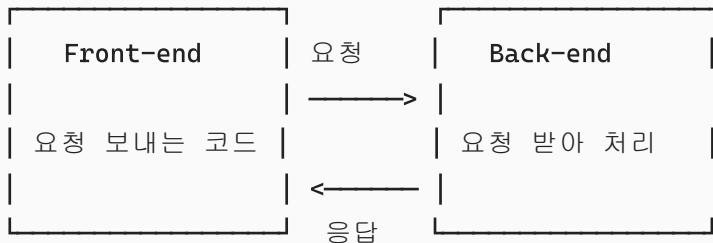
API 종류

1 Internal API (Private API)

개념

Front-end와 Back-end 간 상호작용에 사용되는 API

역할 구분



특징




- 내부 서비스 전용
- 외부에 공개되지 않음
- 보안성이 높음
- 특정 애플리케이션에 최적화

2 Public API

개념

기업이나 공공기관 혹은 개인이 자신들의 데이터와 서비스를 다른 개발자들이 사용할 수 있도록 공개한 API

제공 주체

-  기업: Google Maps API, Twitter API 등
-  공공기관: 기상청 날씨 API, 공공데이터포털 등
-  개인: 오픈소스 프로젝트의 API 등

활용 예시

서비스	API 예시	활용 사례
Google	Maps API	지도 서비스 통합
YouTube	Data API	동영상 정보 조회
GitHub	REST API	코드 저장소 관리
기상청	날씨 API	실시간 날씨 정보

API 설계 방식

주요 설계 방식들

API 설계 방식

- └ REST API ★ (가장 널리 사용)
- └ SOAP API
- └ GraphQL
- └ gRPC

REST API

- 현재 가장 널리 사용되는 API 방식
- HTTP 프로토콜 기반
- 직관적이고 사용하기 쉬움
- [자세한 내용은 REST API 가이드 참조]

API 보안과 인증

주요 인증 방식

방식	설명	사용 사례
API Key	고유 키로 인증	Public API 접근
OAuth	제3자 인증	소셜 로그인
JWT	토큰 기반 인증	웹 애플리케이션
Basic Auth	사용자명/비밀번호	간단한 인증

API 사용 시 고려사항

✅ 장점

- **재사용성**: 한 번 만들면 여러 곳에서 활용
- **확장성**: 새로운 클라이언트 쉽게 추가
- **유지보수**: 로직을 중앙에서 관리
- **표준화**: 일관된 인터페이스 제공

⚠️ 주의사항

- **네트워크 의존성**: 인터넷 연결 필요
- **보안**: 적절한 인증/인가 체계 필요
- **성능**: 네트워크 지연시간 고려
- **버전 관리**: API 변경 시 호환성 유지

🎯 API 선택 가이드

😬 어떤 API를 써야 할까?

프로젝트 특성에 따른 선택

- └ 간단한 웹 서비스 → REST API
- └ 실시간 데이터 필요 → GraphQL + Subscription
- └ 엔터프라이즈급 보안 → SOAP
- └ 고성능 마이크로서비스 → gRPC

📊 학습 순서 추천

1. API 기본 개념 이해 ✅
2. REST API 학습 및 실습
3. GraphQL 개념 학습
4. 실제 프로젝트 적용

[REST API](#)