1장 Modern Web Development

1.1. 서비스와 API

현대 웹은 여러 서비스가 서로 통신하는 방식. 서비스는 사용자 프로필 조회, 결제 처리 등 특정 작업을 수행하는 프로그램.

API(Application Programming Interface)는 서비스 간 통신을 돕는 다리 역할. 마치 식당의 메뉴판과 같음. 내부 동작을 몰라도 메뉴와 주문 방법을 알면 됨. API는 소프트웨어 구성 요소 간 통신 규칙과 프로토콜을 정의.

1.1.1. API의 종류 (프로토콜과 데이터 형식)

API는 통신 방식(프로토콜)과 데이터 구조(데이터 형식)에 따라 분류.

- 프로토콜(통신 규칙):
    - **SOAP:** XML을 사용하는 구식 프로토콜. 주로 기업 환경에서 사용.
    - **REST:** 최신 웹 API의 표준. 유연하고 가벼우며, 표준 HTTP 메서드 사용.
- 데이터 형식(콘텐츠 구조):
    - **XML:** 태그를 사용하는 구조화된 형식.
    - **JSON:** 웹 API의 사실상 표준. 사람이 읽기 쉽고 가벼움.

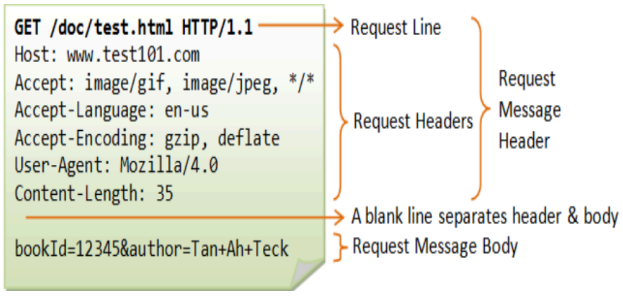
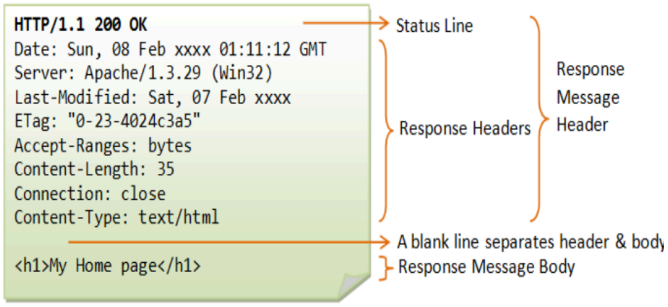
1.1.2. HTTP (HTML, HTTP, URL)

현대 API 이해를 위한 핵심 기술. (팀 버너스리)

- **HTML:** 웹 콘텐츠의 구조를 잡고 표현하는 언어.
- **URL:** 웹상 자원의 고유 주소. 무엇을 찾는지 지칭.
- **HTTP:** 웹에서 데이터를 전송하는 규칙. 서버에 무엇을 할지 명령하는 "동사" 역할.

- **FastAPI** 기능
  - Endpoints 생성이 목적
    - Endpoints :API가 서버에서 리소스(데이터)에 접근할 수 있도록 가능하게 하는 URL

[Client < ----------HTTP Method -------- > Server]

Client	**Server (Data, Server)**
	
★ Client와 Server가 주고 받는 것은 Header 와 Body(click F12 @ any URL) ★ FastAPI로 Python에서 정의한 데이터 Type을 주고 받게 하면 됨	
• GET: 데이터 검색 . 서버에서 특정 리소스를 요청	**Path, Query** GET /customers GET/products/?name=맥북
• POST: 새 데이터 생성 . 서버에 새로운 리소스를 추가 (데이터는 주로 요청 본문에 포함)	**Body (데이터, JSON, Dict)** POST /customers/create
• PUT: 리소스 전체 수정 . 기존 리소스 전체를 새로운 데이터로 덮어	PUT /customers/update
• Patch: 리소스 일부 수정 . 기존 리소스의 특정 필드만 수정	PATCH /products/pitch
• DELETE: 데이터 삭제 . 특정 리소스를 서버에서 제거	DELETE /products/delete

  - Pydantic, Typing, Jinja Template
  - Postrgres (Persistency)
  - Router 등의 개념도 이해하면 ~~~

1.1.3. REST(ful)

REST는 웹 API 구축 아키텍처 스타일. **RESTful** API는 REST 원칙을 따르는 API. 모든 것을 자원으로 간주하고 표준 HTTP 메서드로 상호작용. 직관적이고 사용하기 쉬움.

RESTful URL 예시: `https://api.example.com/users/123`.

- `users`: 사용자 모음
- `123`: 특정 사용자 ID

1.2. 동시성

동시성은 시스템이 여러 요청을 동시에 처리하는 능력. 수많은 사용자에게 빠르고 반응적인 서비스를 제공하는 데 필수적.

- 지연성**(Latency):** 요청 후 응답까지 걸리는 시간. 대기 시간.
- 처리량**(Throughput):** 단위 시간당 처리 가능한 요청 수. 시스템의 용량.

동시성이 높으면 낮은 지연성과 높은 처리량 유지.

1.3. 계층

현대 웹 애플리케이션은 관리와 확장 용이성을 위해 계층 구조로 설계.(MVC)

View

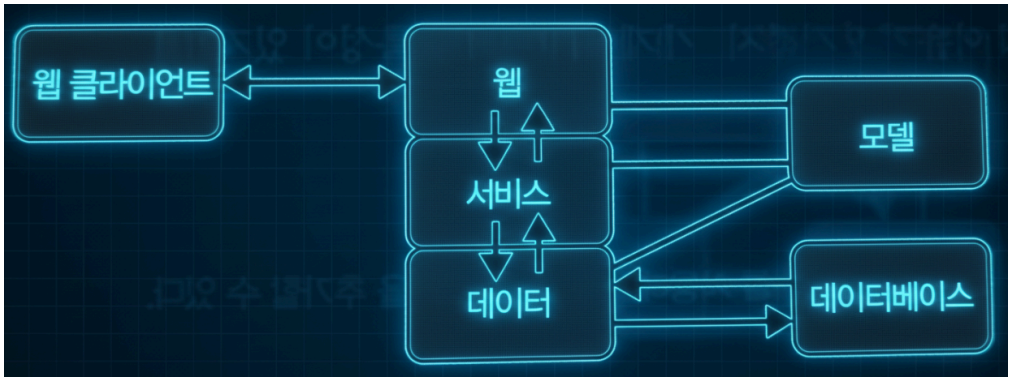
- 웹 클라이언트**:** 웹 브라우저 또는 기타 HTTP 클라이언트 측 소프트웨어
- 웹**:** 클라이언트의 요청을 수집하고, 서비스 계층을 호출해 응답을 반환하는 HTTP를 통한 입력/출력 계층

Controller

- 서비스**:** 필요할 때 데이터 계층을 호출하는 비즈니스 로직(Controller)

Model

- 데이터**:** 데이터 저장소 및 기타 서비스에 접근 : 책
- 모델**:** 모든 계층이 공유하는 데이터 정의 : 분류체계
- 데이터베이스**:** 데이터 저장소 (주로 SQL이나 NoSQL 서버) : 도서관



1.4. 데이터

API 맥락에서 데이터는 클라이언트와 서버 간 교환되는 정보. HTTP 요청/응답의 페이로드.(payload)

- 요청 데이터**:** 클라이언트에서 서버로 보냄. 새 사용자 생성 시 이름, 이메일, 비밀번호 등.
- 응답 데이터**:** 서버에서 클라이언트로 보냄. 사용자 프로필 조회 시 상세 정보 등.

JSON은 데이터에 가장 흔히 사용되는 형식. 간단한 텍스트 기반 형식.

```
JSON
{
  "id": 123,
  "name": "John Doe",
  "email": "john.doe@example.com"
}
```

계층 간 데이터 흐름과 구조 이해는 효율적인 웹 애플리케이션 구축에 매우 중요. 다음 강의는 이러한 웹 계층을 효율적으로 구축하는 **FastAPI**에 대한 내용.