1️⃣

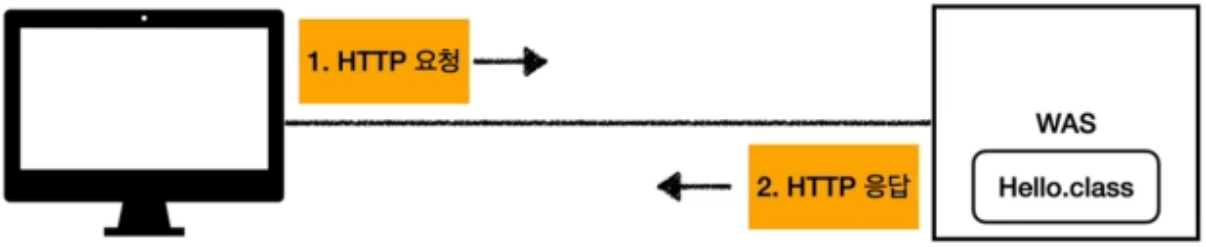
Section1. 웹 애플리케이션 이해

웹 서버, 웹 애플리케이션 서버

웹 서버(Web Server)

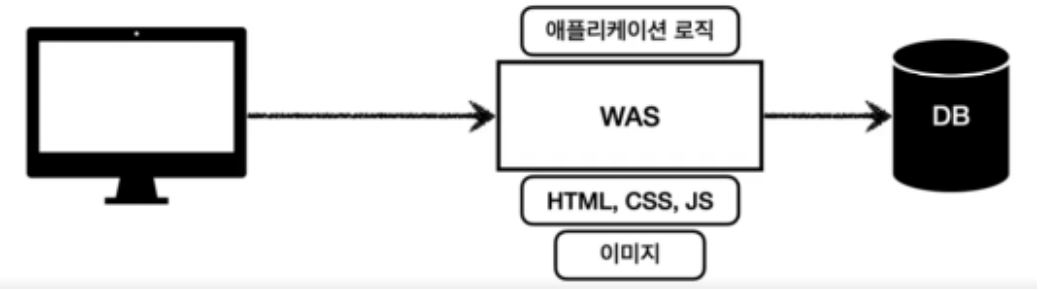
- HTTP 기반으로 동작
- **정적** 리소스 제공
- NGINX, APACHE 등

웹 애플리케이션 서버(WAS)



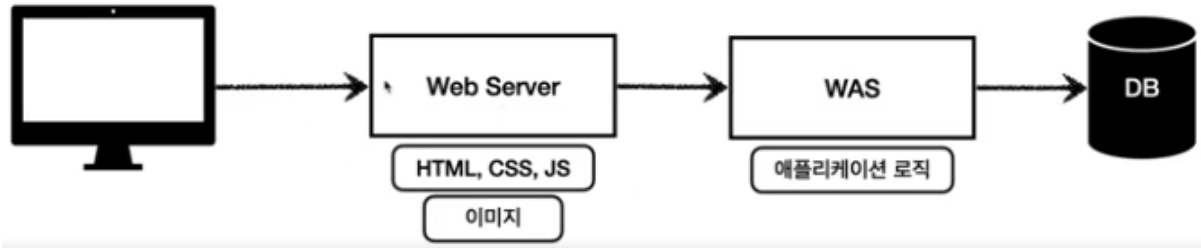
- HTTP 기반으로 동작
- 웹 서버 기능 포함
- 프로그램 코드 실행해서 App 로직 수행
  - **동적** HTML, HTTP API(JSON)
  - 서블릿, JSP, 스프링 MVC
- Tomcat 등

웹 시스템 구성 - WAS, DB



- WAS 가 너무 많은 역할 담당, 서버 과부하 우려

웹 시스템 구성 - WEB, WAS, DB



- **정적 리소스는 웹 서버**가 처리
- 웹 서버는 로직같은 동적인 처리가 필요하면 WAS 에 요청을 위임
- **WAS는 로직 처리** 전담

> 💡 효율적인 리소스 관리
> → 사용되는 리소스에 따른 서버 증설

서블릿

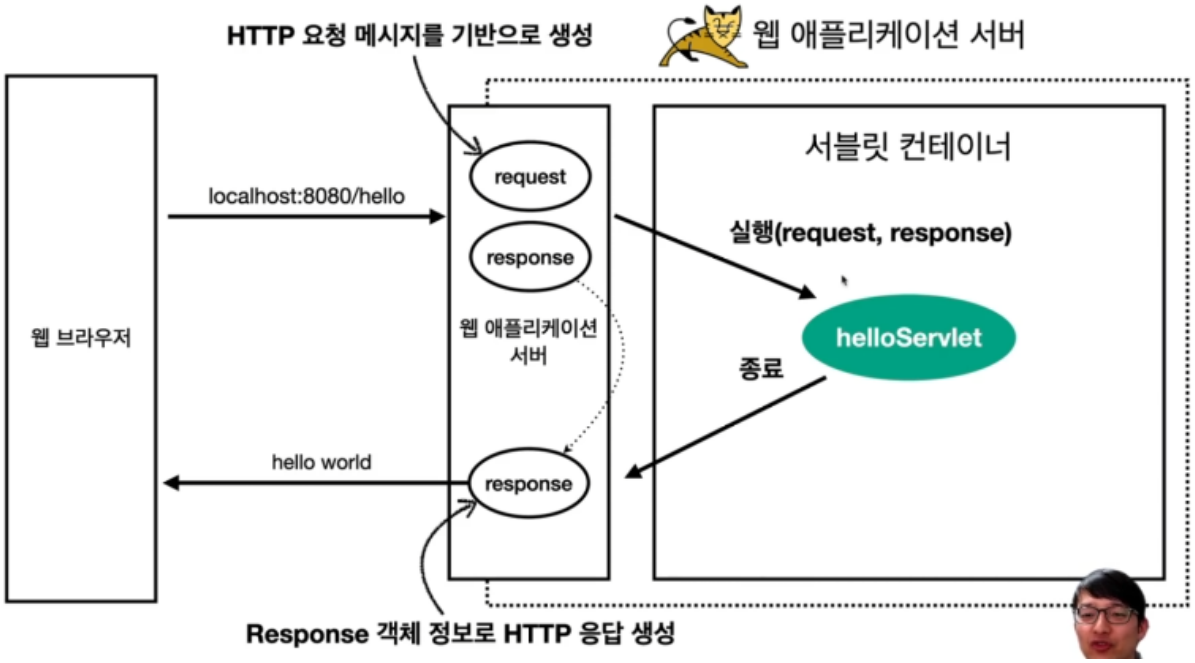
특징

```
@WebServlet(name = "helloServlet", urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response){
      //애플리케이션 로직
    }
}
```

- `urlPatterns(/hello)`의 URL이 호출되면 서블릿 코드 실행
- `HttpServletRequest`, `HttpServletResponse`

HTTP 요청, 응답 흐름



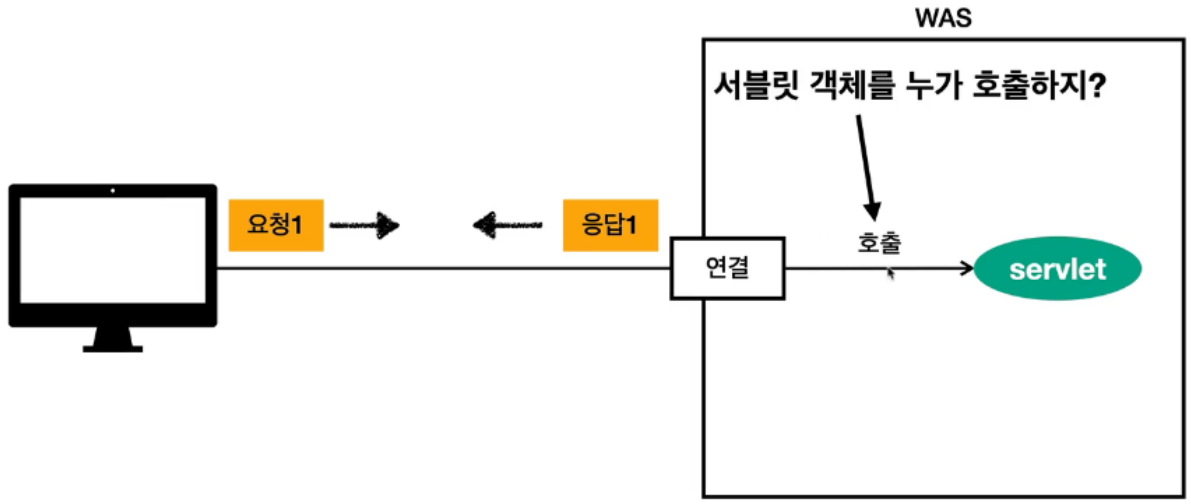
HTTP 요청시

- WAS 는 `req`, `res` 객체를 새로 만들어 서블릿 객체 호출
  - 개발자는 `req` 객체에서 HTTP 요청 정보를 꺼내서 사용
  - 개발자는 `res` 객체에서 HTTP 응답 정보를 입력
- WAS 는 `res` 객체에 담겨있는 내용으로 HTTP 응답 정보 생성

서블릿 컨테이너

- 톰캣과 같은 WAS 를 **서블릿 컨테이너**라고 한다.
  - 서블릿 객체의 생성, 초기화, 호출, 종료 생명주기 관리
- **싱글톤으로 관리**
  - 최초 로딩 시점에 서블릿 객체를 만들어두고 재활용한다.
  - 공유 변수 사용 주의
- JSP 도 서블릿으로 변환 되어 사용된다.
- 동시 요청을 위한 **멀티 쓰레드 처리** 지원
  - 동시에 100명 1000명 지원을 처리하는 이유

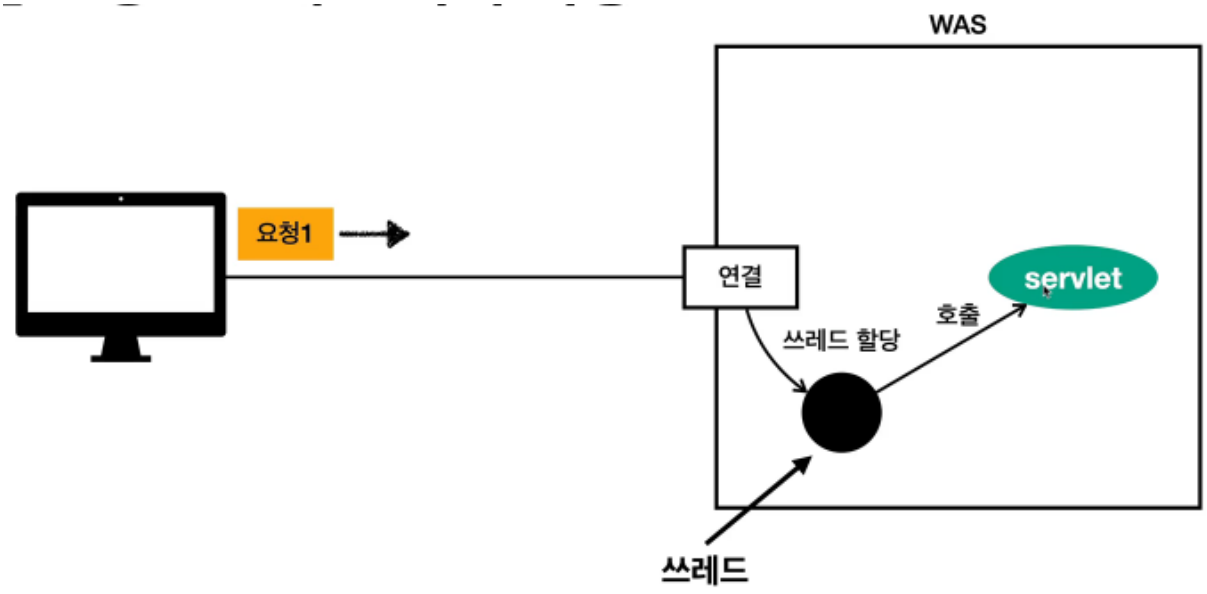
동시 요청 - 멀티 쓰레드



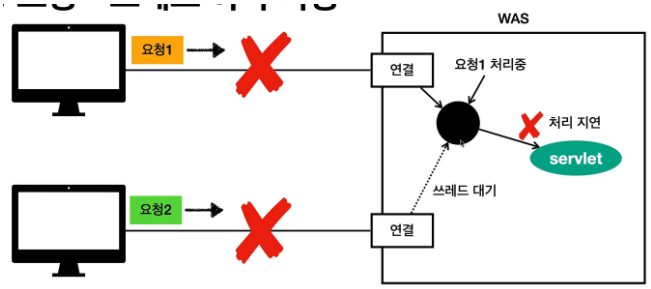
쓰레드

- App 코드를 하나하나 순차적으로 실행하는 것은 **쓰레드**
- java main 메서드는 main이라는 이름의 쓰레드가 실행
- 쓰레드가 없다면 자바 app 실행이 불가능
- 쓰레드는 한번에 하나의 코드 라인만 수행
- 동시 처리가 필요하면 쓰레드를 추가로 생성

단일 요청 - 쓰레드 하나 사용

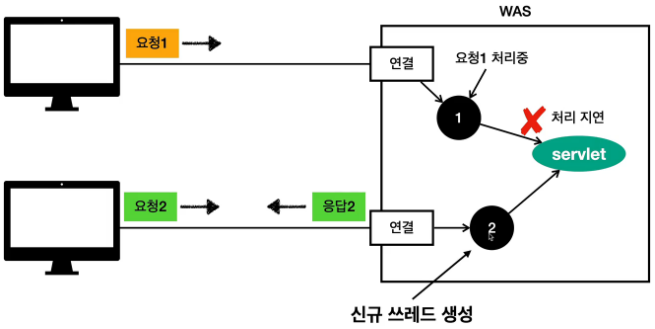


다중 요청 - 쓰레드 하나 사용



- 둘 다 `timeout` 발생 가능성이 있다.

요청 마다 쓰레드 생성



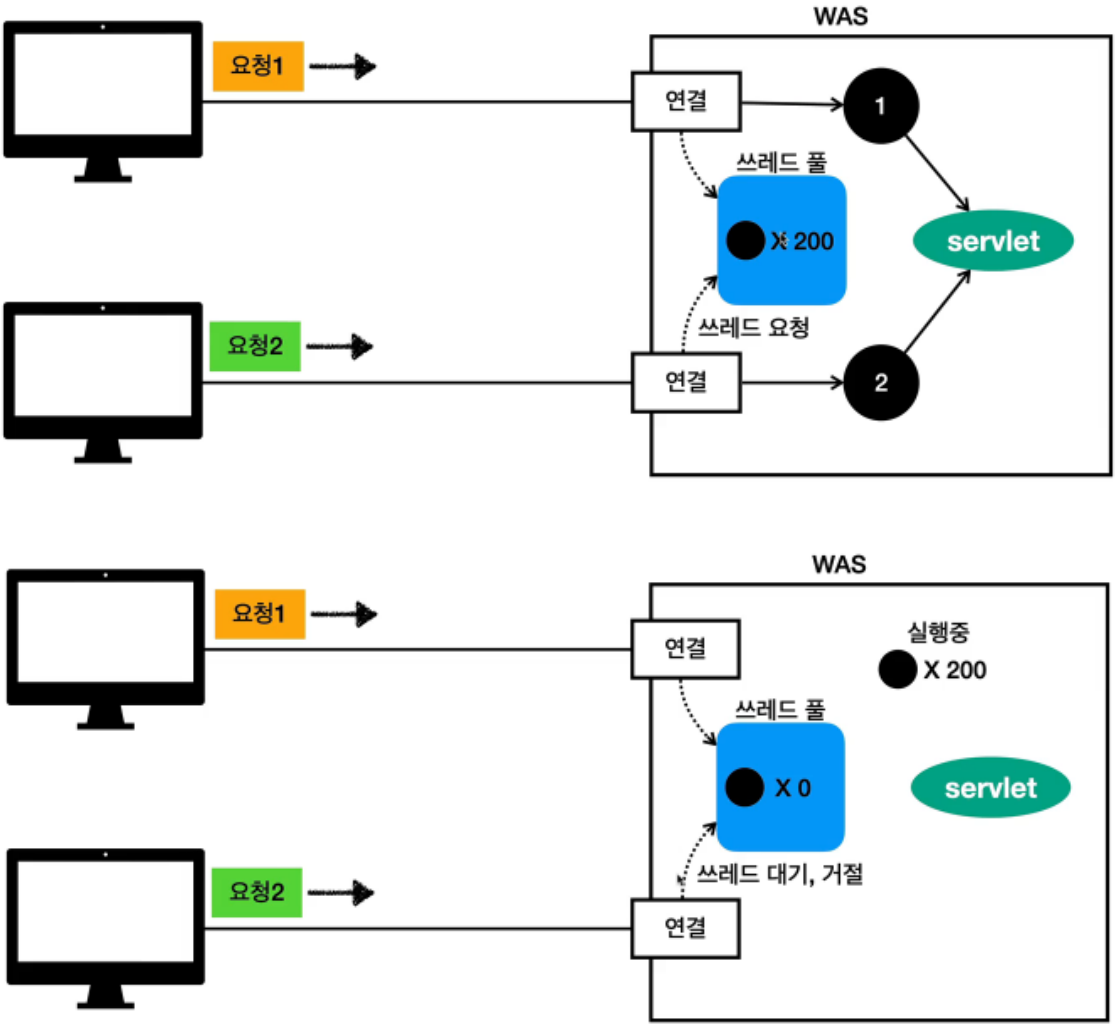
장점

- 동시 요청을 처리할 수 있다.
- 리소스가 허용할 때 까지 처리가능
- 하나의 쓰레드가 지연 되어도, 나머지 쓰레드는 정상 작동한다.,

단점

- 쓰레드는 생성 비용이 매우 비싸다.
  - 고객의 요청이 올 때 마다 쓰레드를 생성하면, 응답 속도가 늦어진다.
- 쓰레드는 **컨텍스트 스위칭 비용이 발생**한다.
- 쓰레드 생성에 제한이 없다.
  - 고객 요청이 많이 오면, CPU, 메모리 임계점을 넘어 서버가 죽을 수 있다.

쓰레드 풀



특징

- 필요한 쓰레드를 쓰레드 풀에 보관하고 관리한다.
- 생성 가능한 쓰레드의 최대치를 관리, 톰캣은 최대 200개 기본 설정

사용

- 쓰레드 필요 시, 이미 생성되어 있는 쓰레드를 풀에서 꺼내서 사용한다.
- 사용을 종료하면 쓰레드 풀에 해당 쓰레드를 반납

장점

- 쓰레드가 미리 생성되어 있으므로, 쓰레드 생성, 종료 비용이 절약되고, 응답 시간이 빠르다.
- 생성 가능 쓰레드의 최대치가 있으므로 많은 요청이 들어와도 기존 요청은 안전하게 처리할 수 있다.

💡 **※ 쓰레드 풀 성능 튜닝 ※**

WAS의 주요 튜닝 포인트는 최대 쓰레드 수이다.

- 이 값을 너무 낮게 설정하면?
  - 동시 요청 시, 서버 리소스는 여유, 클라이언트는 금방 응답 지연
- 이 값을 너무 높게 설정하면?
  - 동시 요청 시, CPU, 메모리 리소스 임계점 초과로 서버 다운
- 장애 발생 시?
  - 클라우드면 일단 서버를 늘리고, 이후 튜닝
  - 클라우드가 아닌 경우 열심히 튜닝

쓰레드 풀의 적정 숫자

- 애플리케이션 로직의 복잡도, CPU, 메모리, IO 리소스 상황에 따라 모두 다름
- 성능 테스트
  - 최대한 실제 서비스와 유사하게 성능 테스트 시도
  - 툴: 아파치 ab, 제이미터, nGrinder

WAS의 멀티 쓰레드 지원

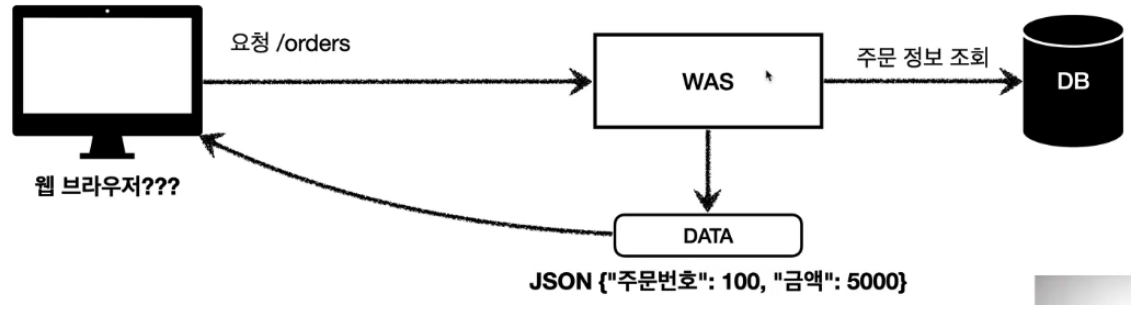
- 멀티 쓰레드에 대한 부분은 WAS가 처리
- 개발자가 멀티 쓰레드 고나련 코드를 신경쓰지 않아도 된다.
- 싱글 쓰레드 프로그래밍을 하듯이 편리하게 소스 코드 개발
- 멀티 쓰레드 환경이므로 싱글톤 객체는 주의해서 사용
  - 멤버 변수, 공유 변수

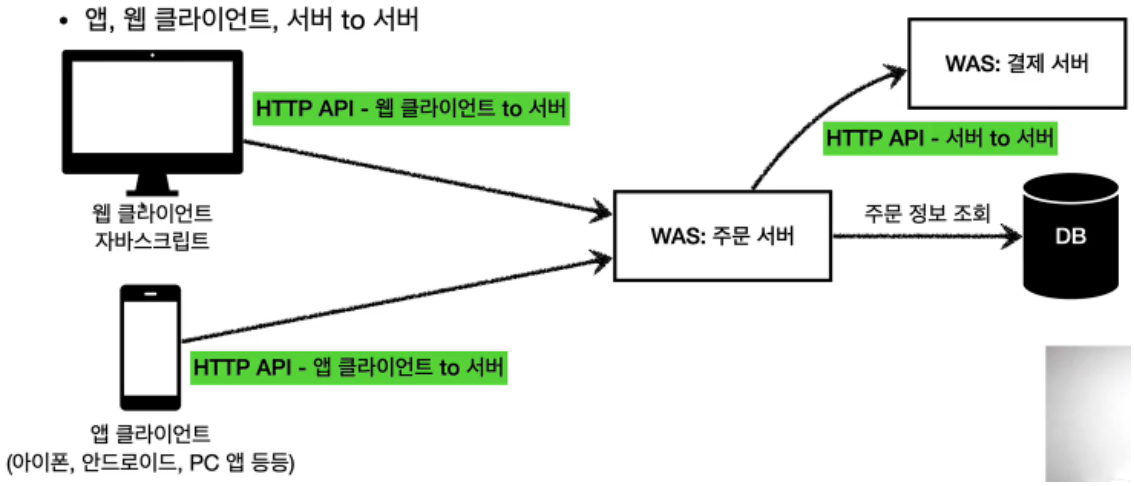
HMTL, HTTP API, CSR, SSR

정적 리소스: 클라이언트 요청 시 서버의 정적 리소스를 반환

HTML: 클라이언트 요청 시 DB 정보 조회 후 동적 HTML 반환

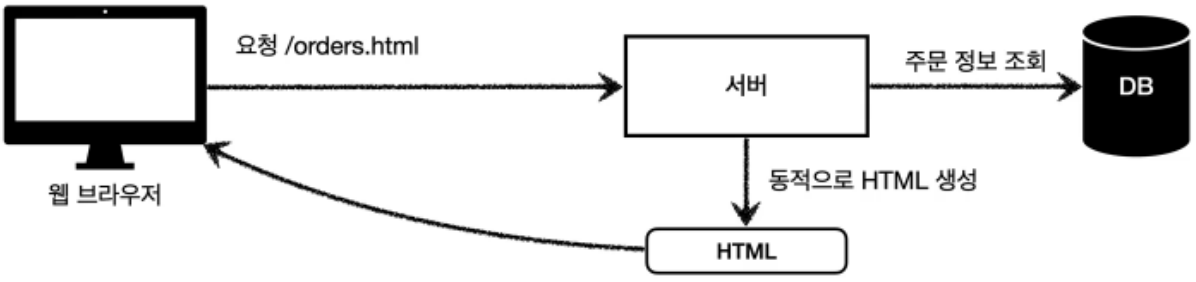
HTTP API





- HTML이 아니라 데이터를 전달
- 주로 JSON 형식 사용
- UI 화면 필요 시, 클라이언트가 별도 처리

서버 사이드 렌더링(SSR)



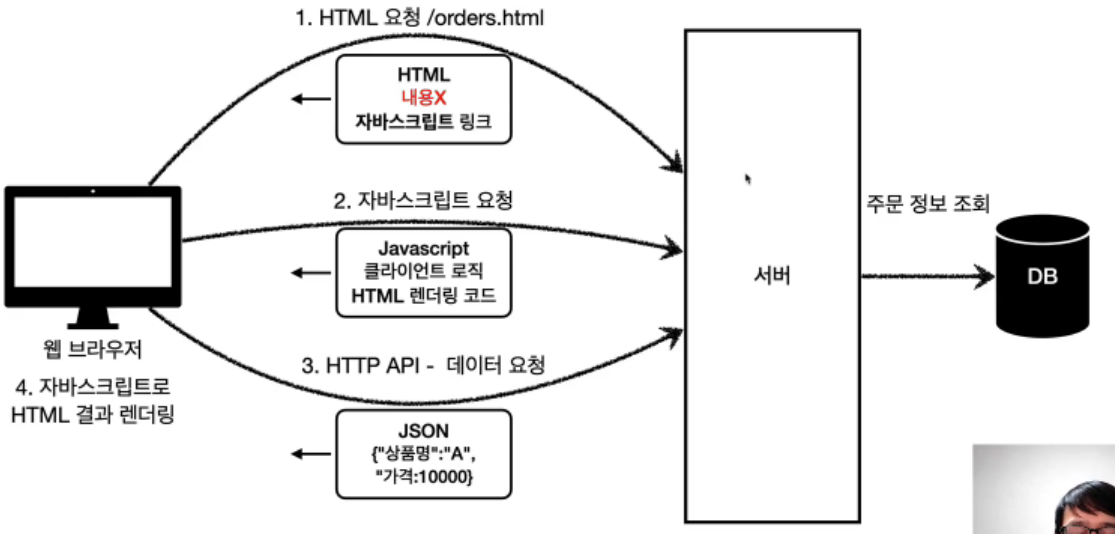
서버에서 최종 HTML을 생성해서 클라이언트에 전달

관련기술

- JSP

- 타임리프

클라이언트 사이드 렌더링(CSR)



관련기술

- React
- Vue.js