**02. 개발에 앞서 알면 좋은 기초 지식**

**1. 서버 간 통신**

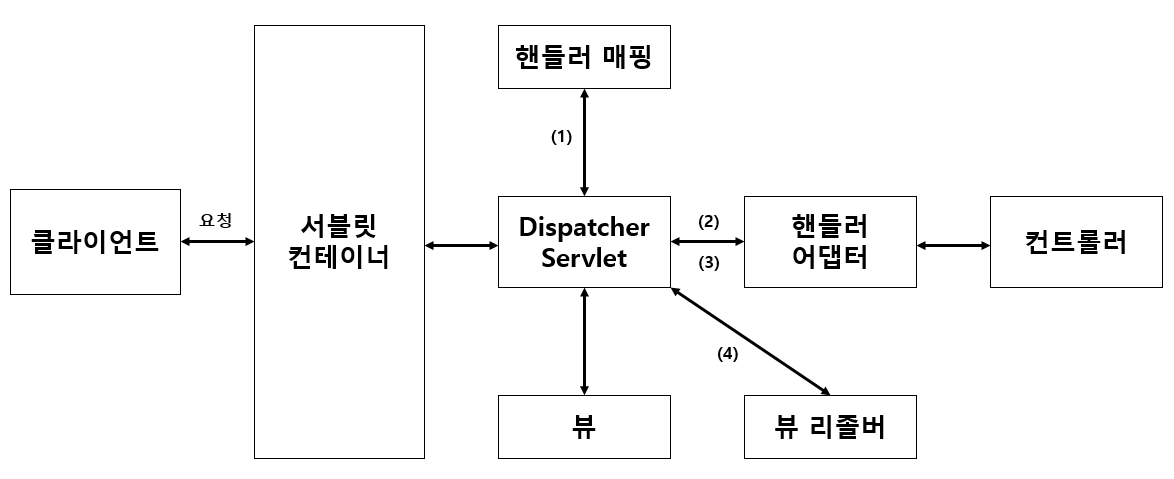
**①** 단일 서비스 아키텍처

: 한 어플리케이션에 모든 기능을 하나로 통합한 구조

**②** 마이크로서비스 아키텍처

: 서비스를 쪼개어 한 어플리케이션을 구현하는 구조

**2. 스프링 부트의 동작 방식**

** ①** 서블릿 컨테이너(Servlet Container)

: 클라이언트의 요청을 처리하고 결과를 반환하는 자바 웹 프로그래밍 기술인 서블릿을 관리 하는 클래스

**특징** – 서블릿 객체를 생성, 초기화, 호출, 종료하는 생명주기 관리

– 서블릿 객체는 싱글톤으로 관리됨

– 멀티 쓰레딩 지원

**②** Dispatcher Servlet

(1) 요청이 들어오면 핸들러 매핑을 이용해 요청 URI에 매핑된 핸들러 검색

(2) 핸들러 어댑터로 컨트롤러 호출

(3) 핸들러 어댑터에 컨트롤러 응답이 들어오면 ModelAndView로 응답을 가공해 반환

(4) 뷰 형식의 컨트롤러 이용 시 뷰 리졸버를 통해 뷰를 받아 리턴

**③** 핸들러 매핑: 요청 정보를 기준으로 컨트롤러를 선택하는 인터페이스

1) BeanNameUrlHandlerMapping: 빈 이름을 URL로 사용

- 슬래시가 들어가면 매핑 대상 ex) @Bean(“/hi”)

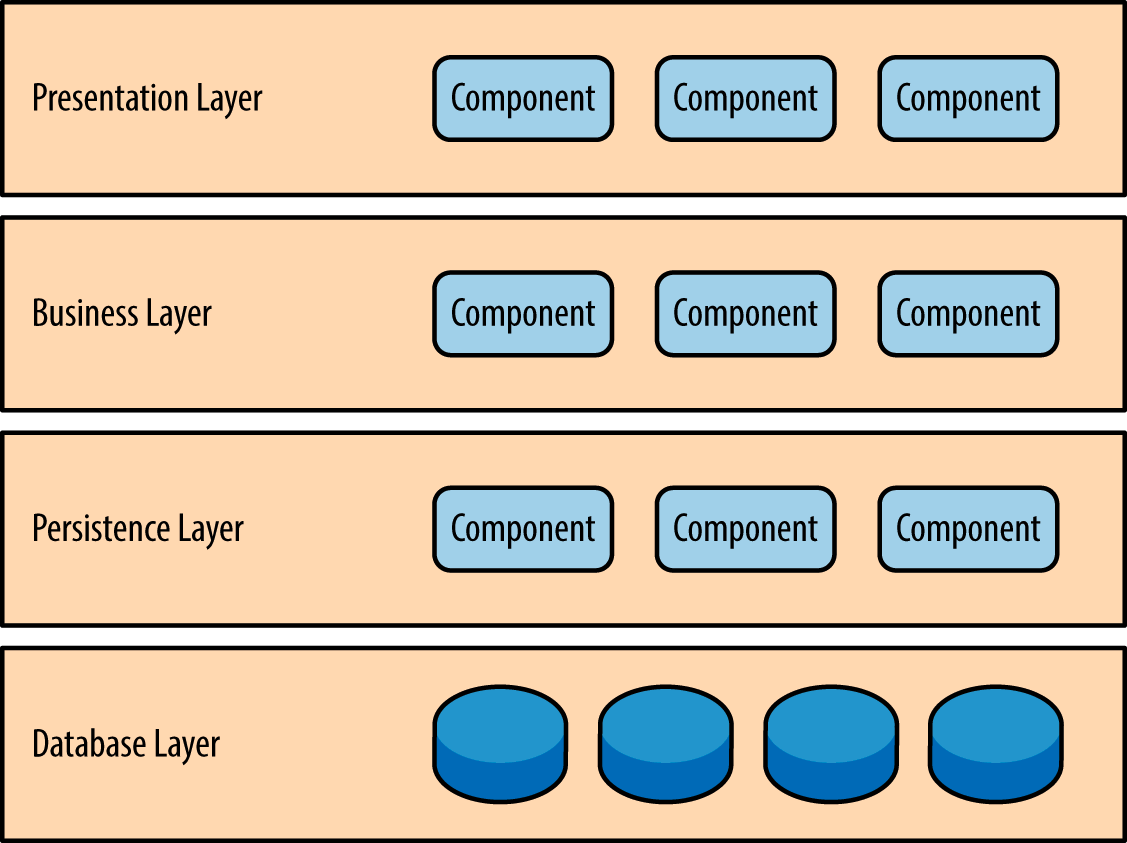
2) ControllerClassNameHandlerMapping

: URL과 일치하는 클래스 이름을 갖는 빈을 컨트롤러로 사용

3) SimpleUrlHandlerMapping: URL 패턴에 매핑된 컨트롤러 사용

4) DefaultAnnotationHandlerMapping: 어노테이션으로 URL과 컨트롤러 매핑

**3. 레이어드 아키텍처**



① 정의: 앱의 컴포넌트를 유사 관심사를 기준으로 레이어드로 묶어 수평적으로 구성한 구조

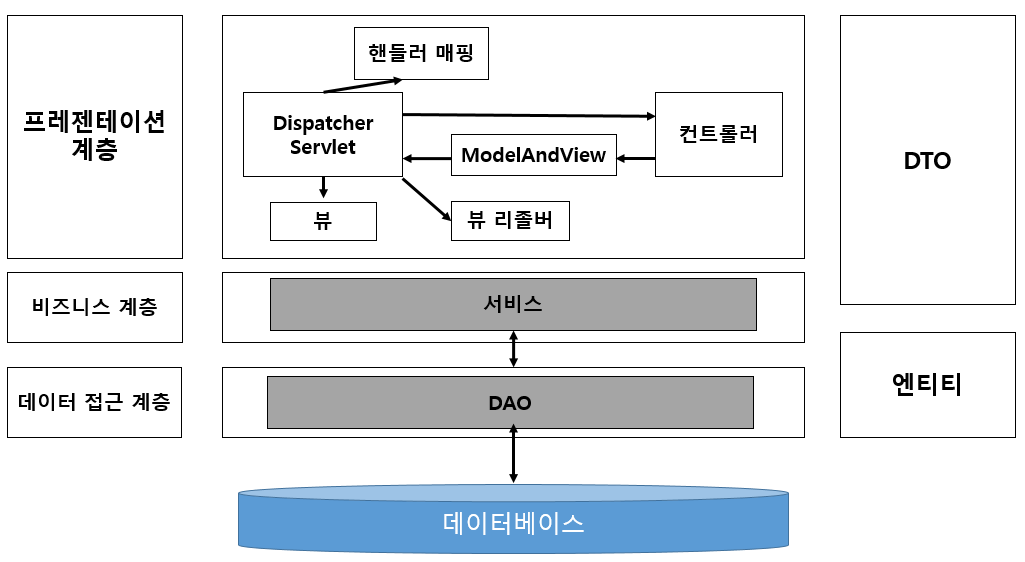
② 계층 설명

1) 프레젠테이션: 클라이언트의 요청을 해석하고 응답. UI 제공

2) 비즈니스: 기능을 정의하고 수행하는 도메인 객체를 역할 수행

3) 데이터 접근: 유에 접근하는 일련의 작업 수행

③스프링의 레이어드 아키텍처



**1)** 프레젠테이션: UI 계층이라고도 하며 클라이언트와의 상호작용 담당

**2)** 비즈니스: 서비스층이라고도 부르며 핵심 비즈니스 로직을 구현하는 영역

**3)** 데이터 접근: DB에 접근하는 작업 수행

**4. 디자인 패턴**

**①** 종류

- 생성 패턴: 객체 생성에 사용되는 패턴

- 구조 패턴: 객체를 조합해서 더 큰 구조를 만듦

- 행위 패턴: 하나의 작업을 여러 객체를 통해 해결

② 생성 패턴

- 추상 팩토리: 구체적인 클래스 지정x. 상황에 맞는 객체 생성을 위해 인터페이스 제공

- 빌더: 객체의 생성과 표현을 분리해 생성

- 팩토리 메서드: 객체 생성을 서브클래스로 분리해서 위임

- 프로토타입: 원본 객체를 복사해 객체를 생성

- 싱글톤: 한 클래스마다 인스턴스를 하나만 생성하도록 함.

③ 구조 패턴

- 어댑터: 클래스의 인터페이스를 의도하는 인터페이스로 변환

- 브리지: 추상화와 구현을 분리하여 독립적으로 변형케 함

- 컴포지트: 복합 객체와 단일 객체를 구별없이 다룸

- 데코레이터: 객체의 결합을 통해 기능을 동적으로 확장

- 퍼사드: 하나의 통합된 인터페이스 제공

- 플라이웨이트: 특정 클래스의 인스턴스 하나로 여러 가상 인스턴스 제공

- 프락시: 특정 객체를 직접 참조하지 않고 대행하는 객체를 통해 접근

④ 행위 패턴

- 책임 연쇄: 요청 처리 객체를 집합으로 만들어 결합을 느슨하게 함.

- 커맨드: 실행될 기능을 캡슐화해 여러 기능을 실행하도록 클래스를 설계

- 인터프리터: 주어진 언어의 문법을 정의하고 해당 언어로 구성된 문장을 해석

- 이터레이터: 내부 구조 노출x. 집합 원소에 순차적으로 접근하게 함.

- 미디에이터: 한 집합의 객체들의 상호작용을 캡슐화함.

- 메멘토: 객체의 상태정보를 저장하고 필요에 따라 상태 복원

- 옵저버: 객체의 상태 변화를 관찰

- 스테이트: 상태에 따라 객체의 행동이 변하도록 설계

- 스트래티지: 행동을 클래스로 캡슐화해 동적으로 행동 바꿈

- 탬플릿 메서드: 특정 작업을 캡슐화해 특정 단계만 변경할 수 있도록 함

- 비지터: 실제 로직을 가지는 객체가 로직을 적용할 객체에 방문 후 실행

**5. REST API**

① REST란?

: Representational State Transfer의 약자로 분산 하이퍼 미디어 시스템 아키텍처. 주고 받는 Resource를 규정하고 URI에 명시하여 HTTP메서드를 통해 주고 받음

② 특징

- 유니폼 인터페이스: HTTP 전송 규약을 따르기 때문에 호환해 사용 가능

- 무상태성: 서버에 상태 정보를 따로 보관하거나 관리하지 않는다.

- 캐시 가능성: HTTP의 캐싱 기능 적용 가능

- 레이어 시스템: 여러 계층으로 구성될 수 있음.

- 클라이언트-서버 구조: API를 제공하고 클라이언트와 서버를 분리해 설계