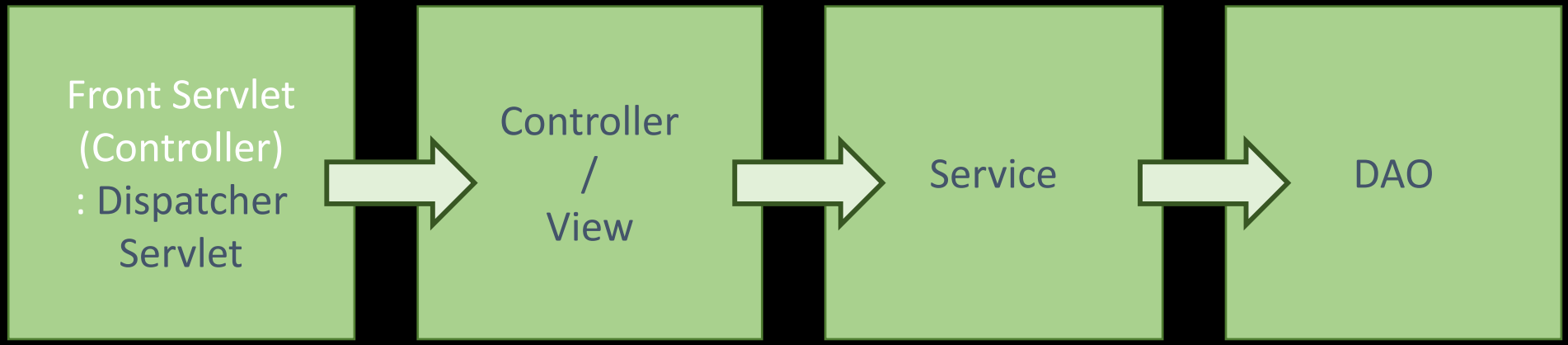
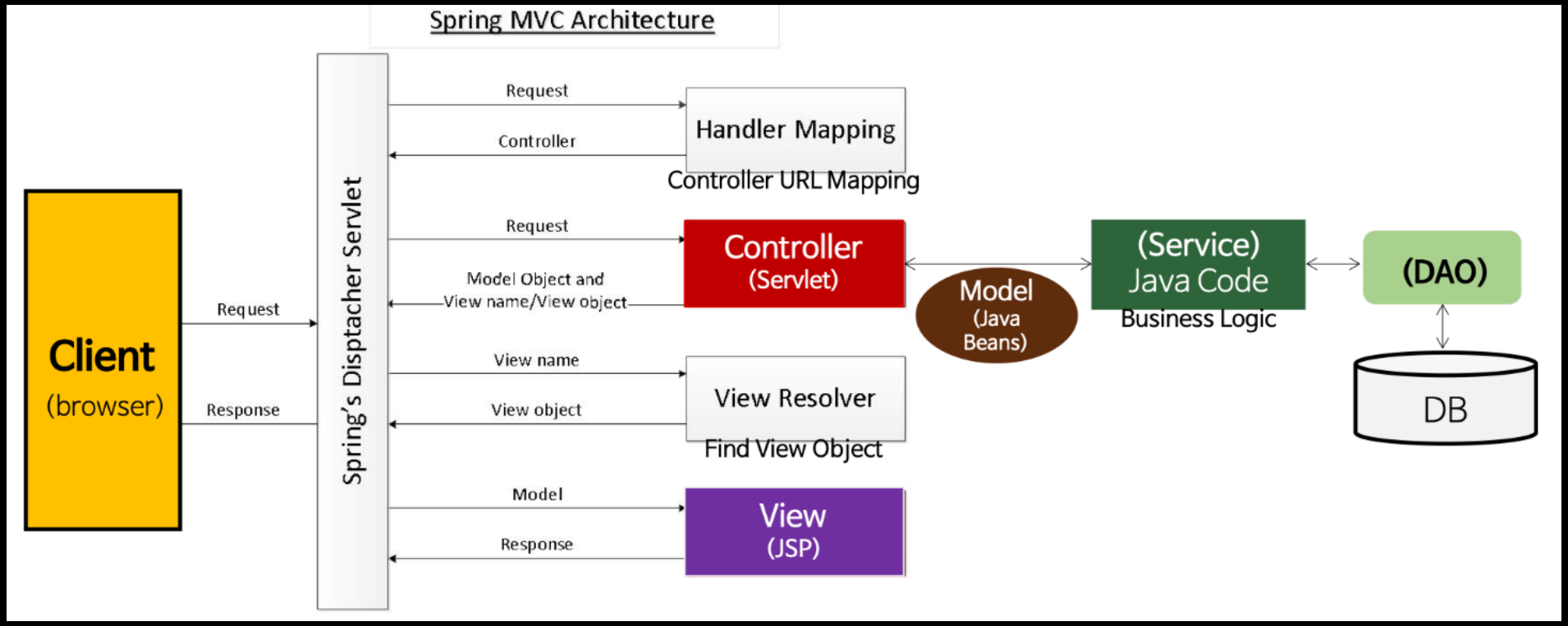
# Chapter 15
## 웹 어플리케이션 구조

정승혜

# 간단한 웹 App의 구성 요소

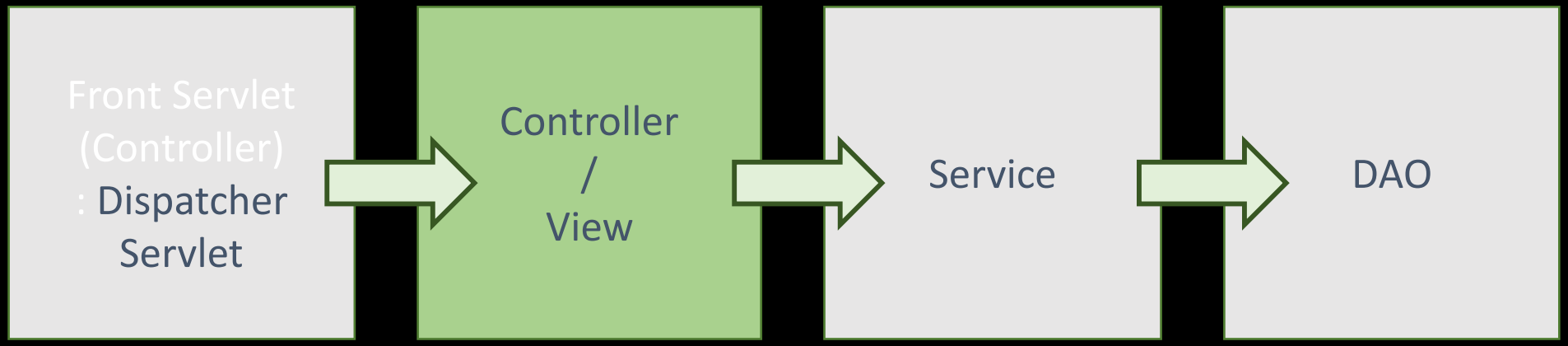


# 간단한 웹 App의 구성 요소




http://gmlwjd9405.github.io/2018/12/20/spring-mvc-framework.html

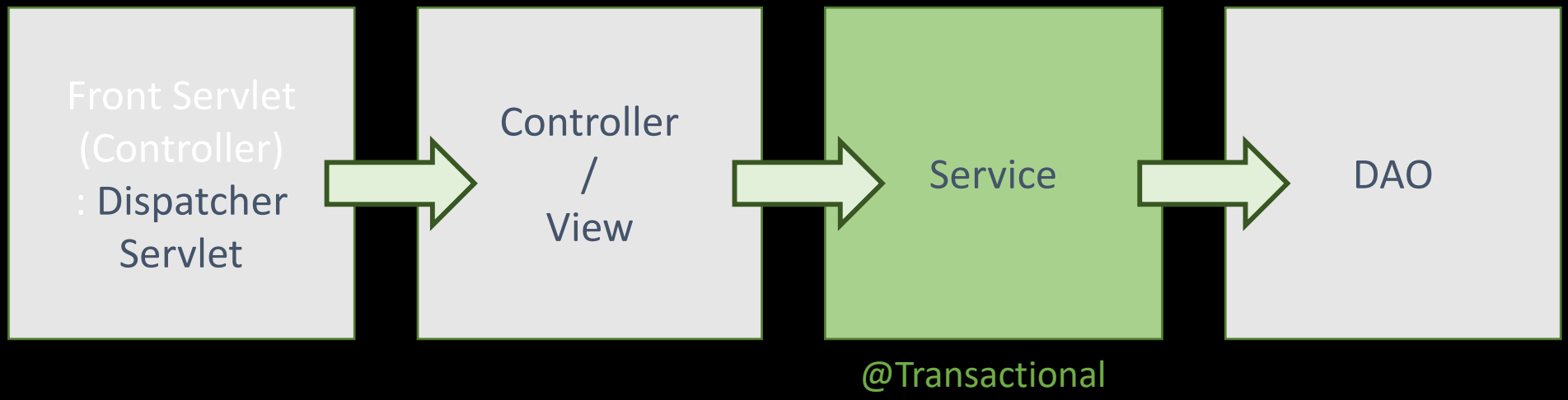
# 간단한 웹 App의 구성 요소




- app이 제공하는 기능과 사용자 요청을 연결하는 매개체
- 로직을 제공하는 서비스에 처리 위임

- 클라이언트가 요구한 기능을 실행
- 응답 결과를 생성하는데 필요한 모델 생성
- 응답 결과를 생성할 뷰 선택

# 간단한 웹 App의 구성 요소

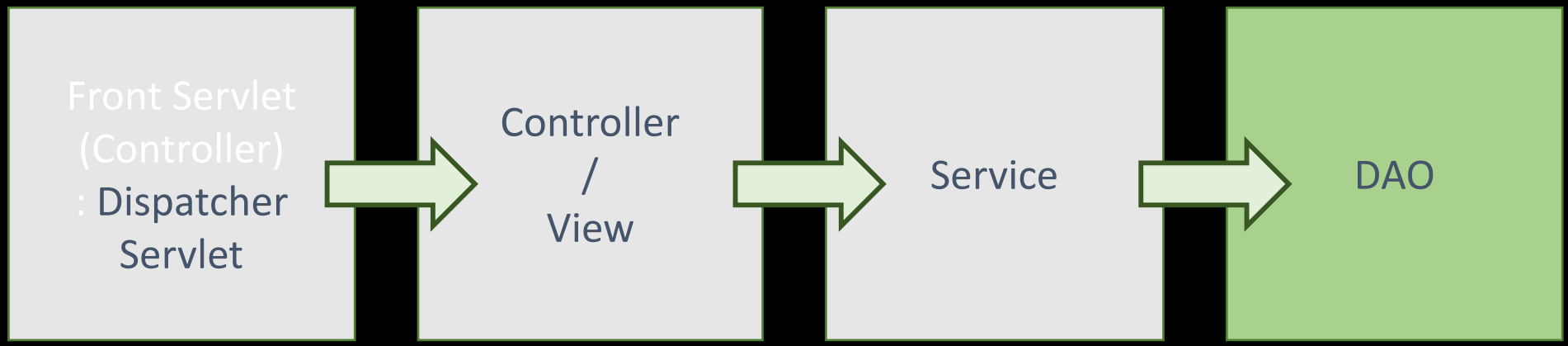



- 로직 구현
  ex ) 수정폼 제공, 로그인 여부 확인, 비밀번호 변경

- View의 폼 값 바인딩/검증, 폼 태그 연동 기능 사용시
  커맨드 클래스를 파라미터로 사용

- 리턴/익셉션을 통해 정상/비정상 결과 리턴 필수

# 간단한 웹 App의 구성 요소



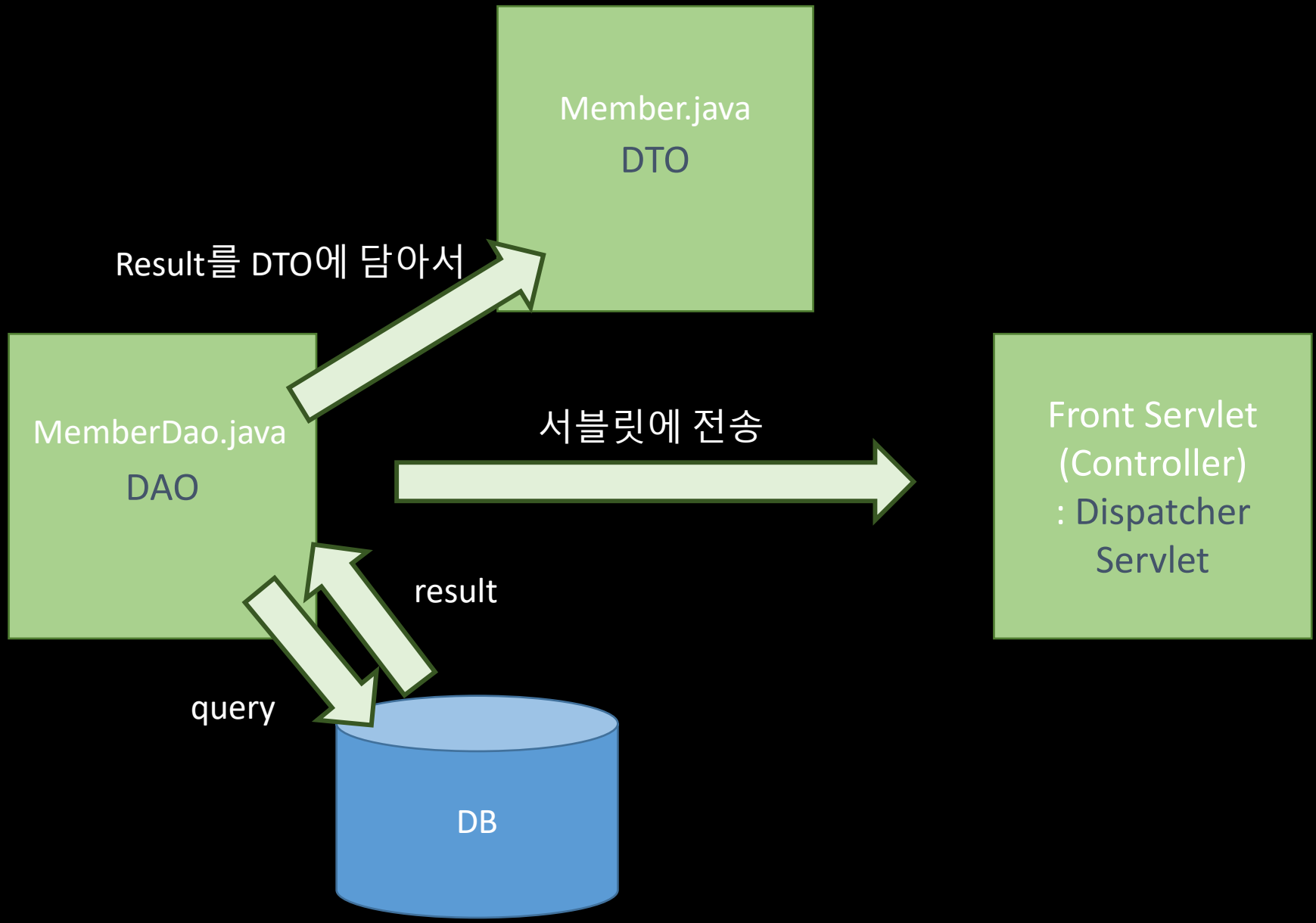

- DB 와 웹 사이에서 데이터를 이동시켜주는 역할

- 목록이나 상세 화면 같이 데이터 조회 기능
  / 부가적인 로직이 없는 경우에는
  컨트롤러에서 직접 DAO를 사용하기도 함

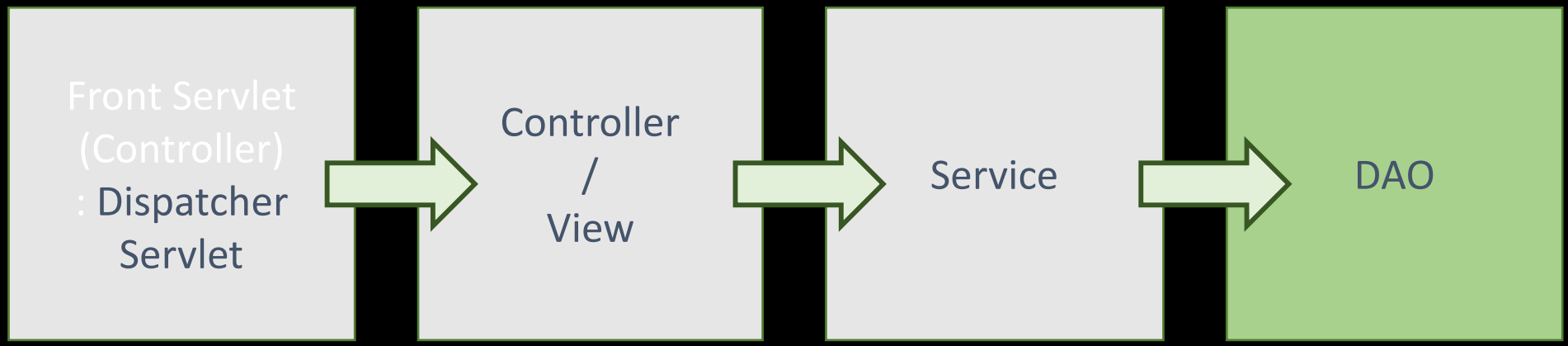
# DAO / DTO / VO

| DAO (Data Access Object ) | DTO ( Data Transfer Object ) |
| --- | --- |
| - Connection Pool에서 특정 user를 위해 선택된 Connection에서 사용하는 데이터베이스 접근 객체<br><br>- Mybatis 등 사용할 경우 커넥션풀이 제공되서 DAO를 별도로 만드는 경우는 드뭄 | - 계층간 데이터 교환을 위한 자바빈즈 DB에 존재하는 테이블의 데이터를 담음<br><br>- 계층간 : Controller, View, Business Layer, Persistent Layer<br><br>- 로직을 가지지 않는 순수한 데이터 객체 getter / setter만 존재 |

| VO ( Value Object ) |
| --- |
| - DTO와 혼용해서 쓰이기도 함<br>- 값 오브젝트으로써 값을 위해 쓰임. ReadOnly, getter 기능만 존재<br><br>Ex ) Color.Red = ~~ |


Member.java
DTO
Result를 DTO에 담아서
MemberDao.java
DAO
서블릿에 전송
Front Servlet
(Controller)
: Dispatcher
Servlet
result
query
DB

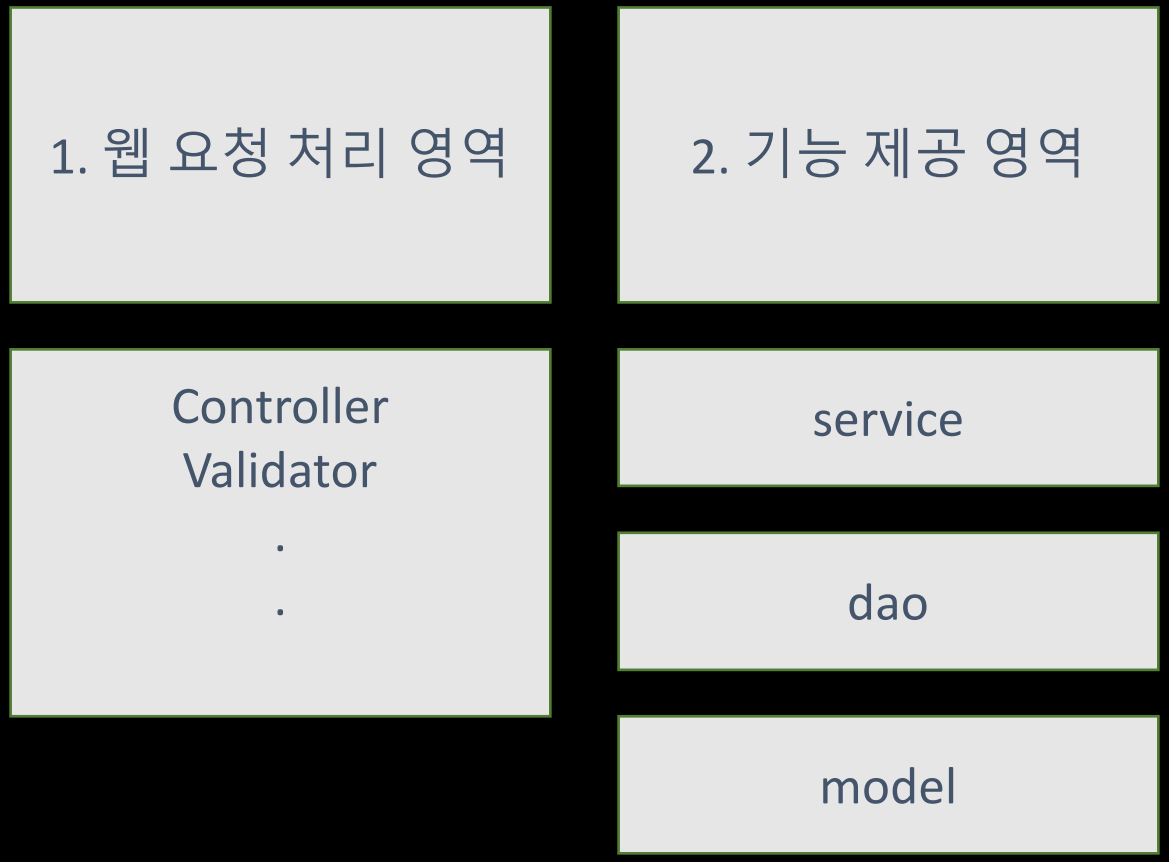
# 간단한 웹 App의 구성 요소




- DB 와 웹 사이에서 데이터를 이동시켜주는 역할

- 목록이나 상세 화면 같이 데이터 조회 기능
  / 부가적인 로직이 없는 경우에는
  컨트롤러에서 직접 DAO를 사용하기도 함

# 패키지 구성과 영역 설계

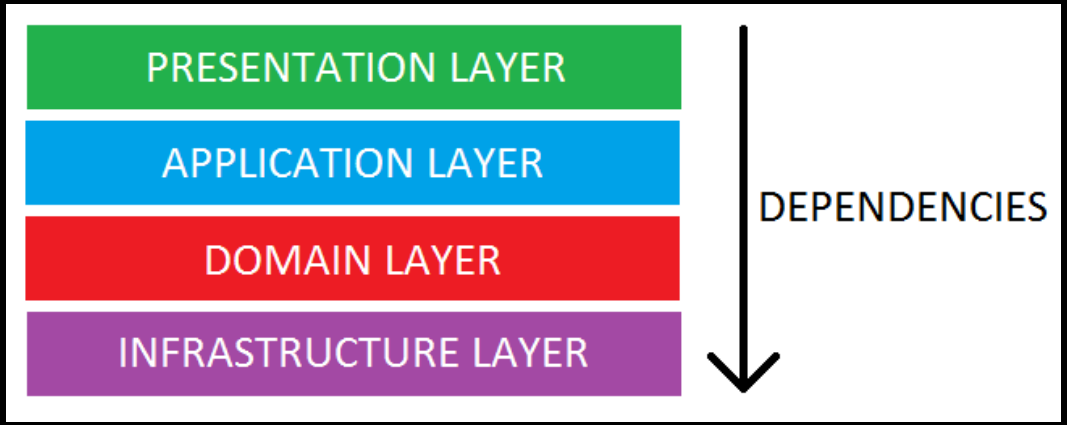


# 도메인 주도 설계 (DDD)

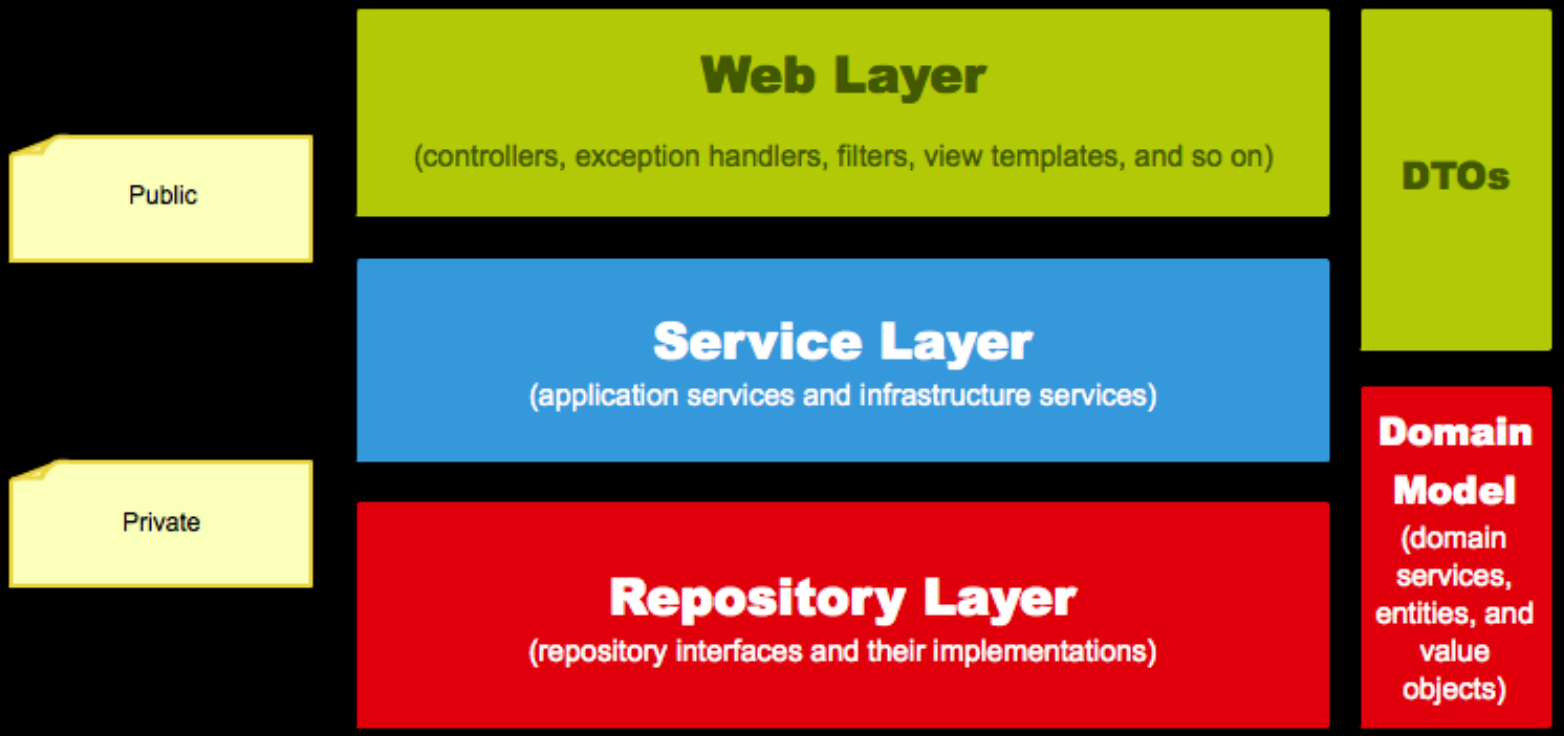
Domain Driven Development
- 데이터에 종속되는 앱/모델링과 개발의 불일치 해결
- 공통의 언어로 도메인과 구현을 충분히 만족하는 모델을 만듦

Ex ) 새로운 요구 사항 도출
1. 출고 전 배송지 변경 가능하게
2. 배송 전에 주문 취소 가능하게

⇒ 각각 출고 , 주문, 배송 Service로 구현하게 되면 유지보수 / 코드 관리 어려움
⇒ Application(Service)를 하나로 따로 빼고, Domain 영역에 각각의 비즈니스 로직 구현



# 도메인 주도 설계 (DDD)




https://www.petrikainulainen.net/software-development/design/understanding-spring-web-application-architecture-the-classic-way/

# 도메인 주도 설계 (DDD)

- **PRESENTATION LAYER** : UI 영역
- 사용자의 요청-> 응용 영역에 전달 / 처리 결과를 다시 사용자에게 보여줌
- (Controller 영역, DispatcherServlet에게 요청과 응답을 전달하는 역할)
- 사용자 세션 관리

- **APPLICATION LAYER** : 응용 영역
- 비즈니스 로직 없음
- 도메인 계층에게 업무 처리 위임 / 조합함 – Service 역할
- 트랜잭션 처리, 도메인으로 부터 이벤트 처리

- **DOMAIN LAYER** : 도메인 영역
- 비즈니스 로직
- 도메인 모델을 구현 (이름, 주소, 상품, 주문서 등)

- **INFRASTRUCTURE LAYER** :
- RDB , MQTT, API.....
- (외부 API, 데이터베이스, 외부 라이브러리 사용 등)