SK 주식회사

KAIST

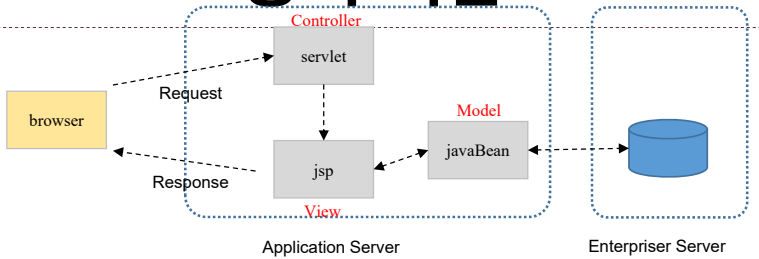
# 각 Layer구조 설계 (Architecture 고려사항)

SK(주) C&C 한정헌

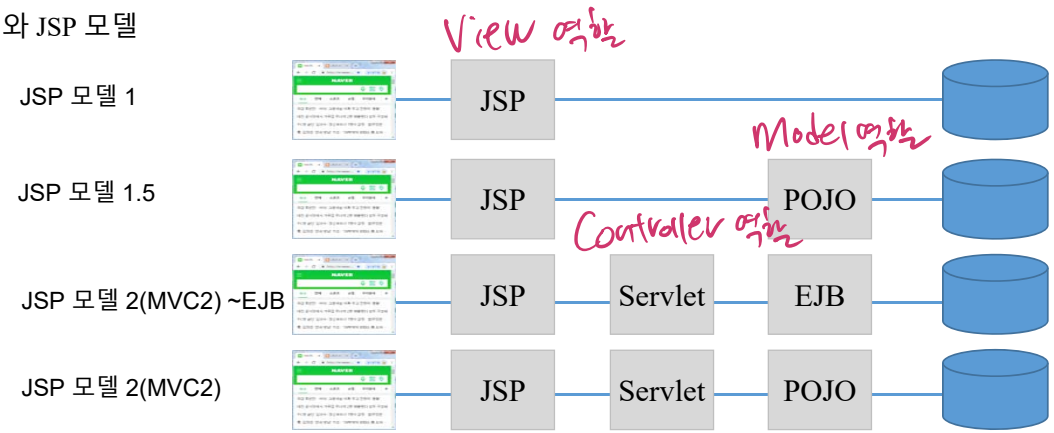
# Presentation 층의 역할

http://javabykiran.blogspot.com/2013/06/mvc2-architecture.html

## MVC2




✓ MVC2 와 JSP 모델



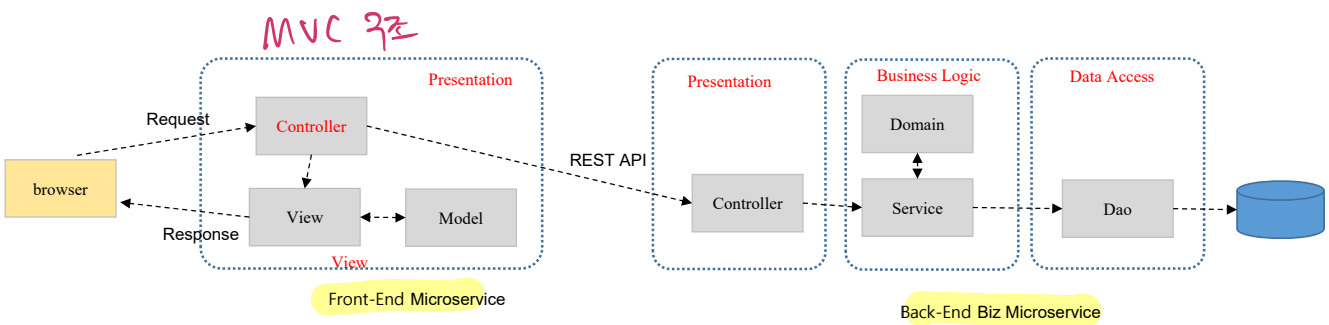
SK 주식회사

KAIST

# Presentation Layer 고려사항

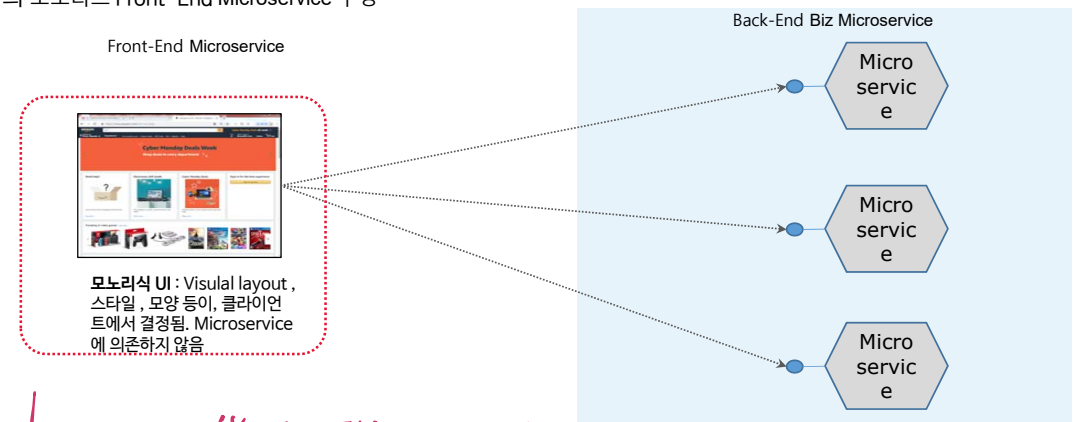
- 사용자 인터페이스 다양화(WEB 브라우저, 모바일등)
- 리치 클라이언트, 리액트, 앵글러, 뷰, 리액트, Spring MVC
- 클라이언트 Tier에서 존재하는 경우도 있음(안드로이드앱,어플)
- Presentation Layer에 어떤 기술을 사용할지 결정한 다음 그에 맞는 설계 고려 필요

# Presentation Layer 의 역할



# Presentation Layer 고려사항

- Client-side UI composition
- 각 팀은 자신의 서비스를 위해 페이지 / 화면의 영역을 구현하는 AngularJS과 같은 Client 측 UI 구성 요소를 하나의 모노리스 Front-End Microservice 구성




↳ 모노리스 시스템의 단점이 나타남

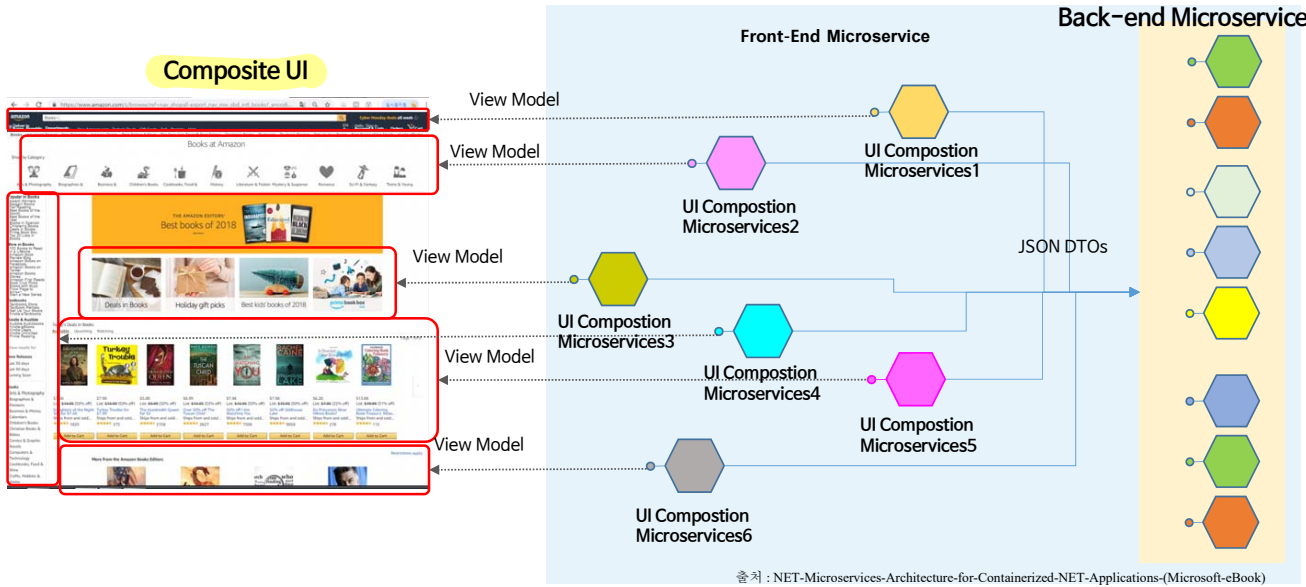
출처 : NET-Microservices-Architecture-for-Containerized-NET-Applications-(Microsoft-eBook)

http://meuslivros.github.io/BUIlding%20Microservices%20-%20Sam%20Newman/text/part0006_split_039.html

⇒ Front end Micro Service로 독립적으로 분해하자

# Presentation Layer 고려사항

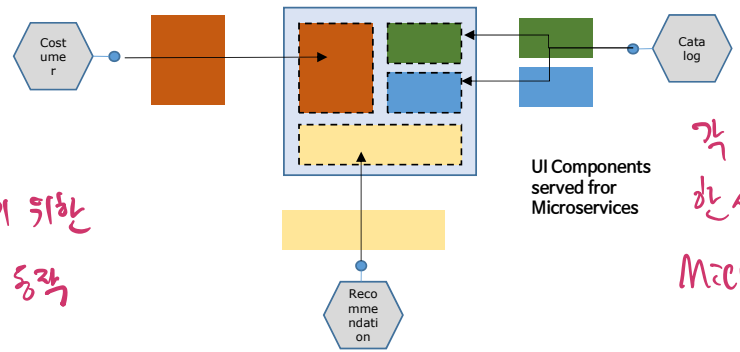
- Server-side page fragment composition




출처 : NET-Microservices-Architecture-for-Containerized-NET-Applications-(Microsoft-eBook)

http://meuslivros.github.io/BUIlding%20Microservices%20-%20Sam%20Newman/text/part0006_split_039.html

# Server-side page fragment composition




서버에는 각 조각을 모으기 위한
간단한 웹 프레임 서버또는 응용

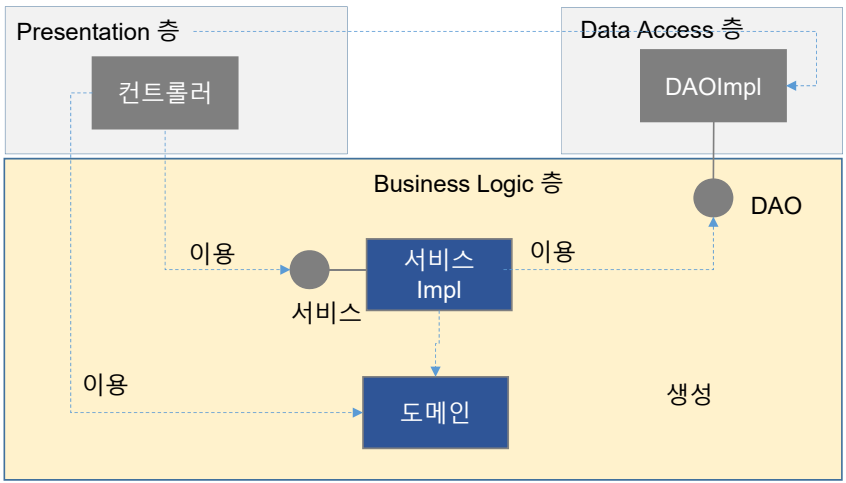
각 조각 영역은 비즈니스를 구성하는
한쌍 이상의 Front-End, Back-End
Micro Service 에 의해 제공

출처 : NET-Microservices-Architecture-for-Containerized-NET-Applications-(Microsoft-eBook)

http://meuslivros.github.io/BUIlding%20Microservices%20-%20Sam%20Newman/text/part0006_split_039.html

# Business Logic 층의 역할

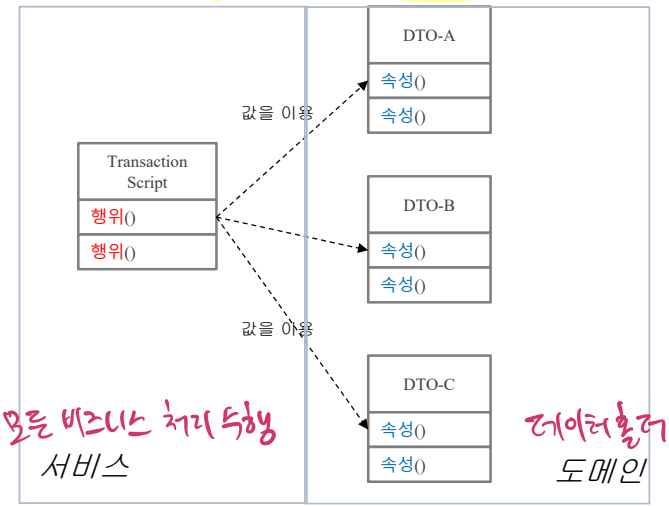
비즈니스 민감성을 처리할 핵심 Layer



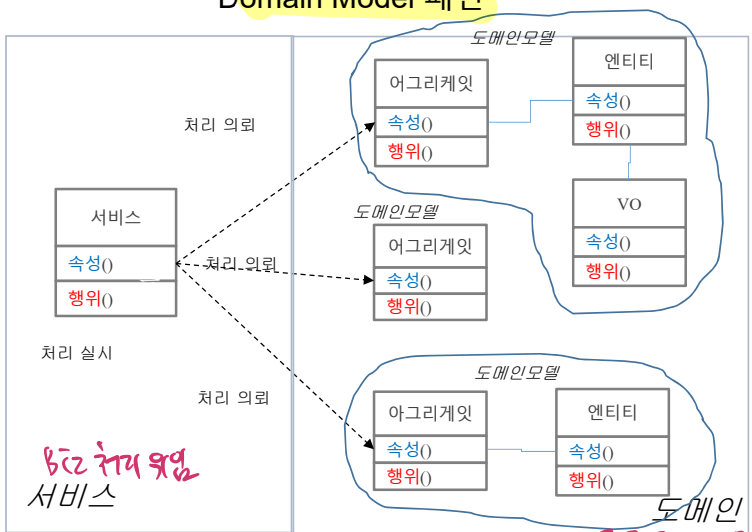
# Business Logic 층의 역할

## Transaction Script VS Domain Model ,

트랜 젝션 스크립트 패턴




Domain Model 패턴




단순 입출력 Application,
객체지향 지식없는 프로그래머가 많이 있는 대규모 프로젝트
서비스가 행위없는 DTO 호출

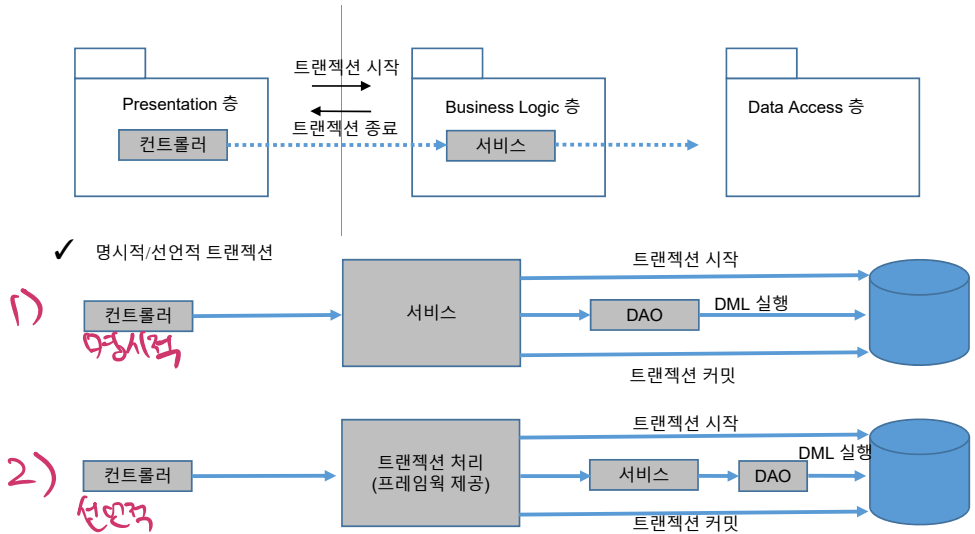
단순 입출력 구조 → Transaction Script Pattern

비즈로직이 복잡한 업무, 객체지향 활용
에릭에반스 DDD 패턴 활용 도메인모델링

복잡한 Biz 로직 → Domain Model Pattern

# Business Logic 층의 Architecture 고려 사항

- Spring DI , AOP
- 트랜젝션 경계



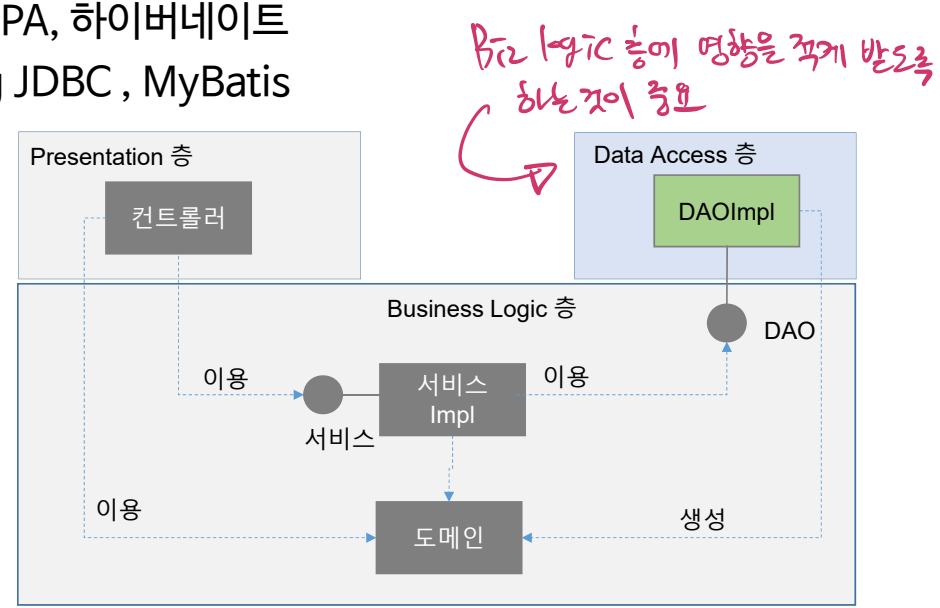

선언적 transaction을 사용하면 Transaction 코드 작업 불필요 → 코드의 간결성, B/L 로직 집중가능

# Data Access 층의 Architecture 고려사항

OR 매핑 VS SQL 매핑

OR 매핑 기술 : Spring JPA, 하이버네이트

SQL 매핑 기술 : Spring JDBC , MyBatis

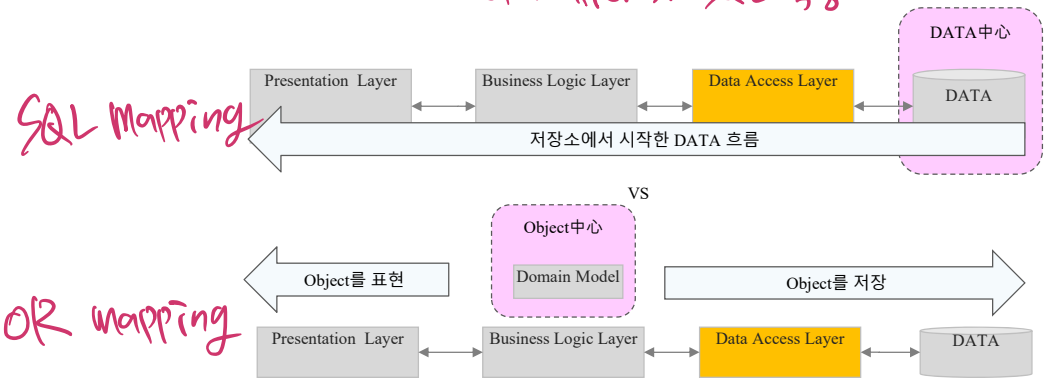


# Data Access 층의 역할

개발자가 직접 SQL 작성

- SQL 매핑 : 데이터 모델링을 통해 테이블을 먼저 작성, 기존 테이블이 이미 존재하는 경우, Transaction Script 패턴에 잘 맞음
- OR 매핑 : RDB 액세스를 Business Logic에서 숨기고 Business Logic에 필요한 데이터를 테이블에서 취득, 오브젝트에 매핑 하는 것. 객체지향 중심, 객체지향 중심으로 엔티티(도메인모델의 클래스)를 추출, 그 엔티티를 바탕으로 테이블 작성

OR Mapper 가 SQL 작성



SK 주식회사

KAIST

# Data Access 층의 Architecture 고려사항

- 비즈니스 성격, 팀원의 역량, 개발 효율성이 고려되어야 함
- 저장소나 RDB가 바뀌어도 비즈 로직에 영향이 미치지 않도록 고려