1. 프레임워크 기초

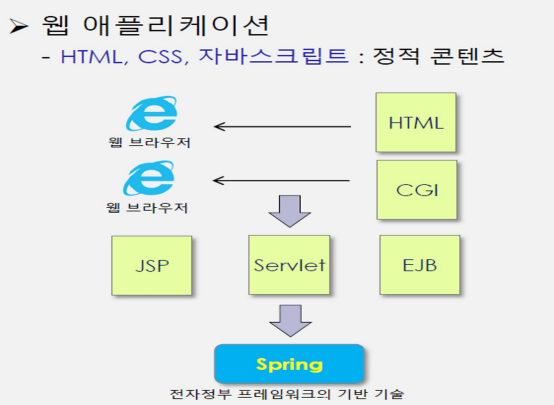
2020년 3월 17일 화요일 오전 11:56

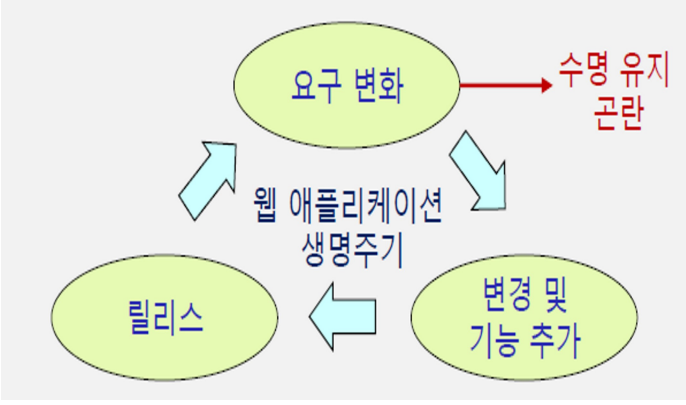
일반적인 웹 시스템

- 정적 컨텐츠(HTML)
  - 웹 브라우저가 웹 서버로부터 저장된 HTML을 읽어와 표현
- 동적 컨텐츠(JSP)
  - 웹 브라우저가 웹 서버에 동적 페이지를 요청하면 이 요청을 애플리케이션 서버가 실행하고, 처리 결과를 브라우저가 HTML형식으로 받아서 표현

웹 애플리케이션

- CGI: 동적 콘텐츠
  - 매번 프로그램 실행으로 성능저하
  - 트랜잭션 관리 어려움
- JSP, Servlet: 웹 컨테이너가 세션 관리
  - 웹 페이지와 비즈니스 로직 분리
  - 자바 언어 사용, 오브젝트 지향의 장점 '재사용'
- EJB(Enterprise JavaBean): 분산 트랜잭션 처리 컴포넌트(분산 환경)
  - JSP, Servlet으로 웹 프레젠테이션 처리
  - EJB 분산 객체로 비즈니스 로직 처리
  - J2EE 버전업과 함께 EJB 복잡도는 높아지고 확장성은 떨어짐
- Spring: DixAOP 컨테이너
  - 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크(사실상의 J2EE의 표준 프레임워크)
  - POJO(Plain Old Java Object) 지원을 통해 기존 EJB의 문제점인 복잡성을 줄임
  - POJO는 특정 인터페이스 또는 클래스를 상속받을 필요가 없는 가벼운 객체
  - 경량 컨테이너, POJO객체
  - 웹, 클라우드, 모바일등 다양한 자바 기반의 애플리케이션을 만들기 위한 프레임워크
  - 대한민국 전자정부 프레임워크의 기반 기술





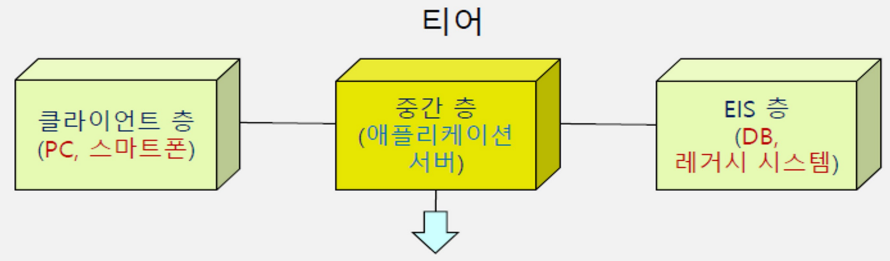
아키텍처의 중요성

- 서버 애플리케이션 구조의 변화

    - Client-Server 구조에서 Web Server 기반 구조로 진화
    - 요구사항 변경, 잦은 기능 추가 발생, 미래 환경의 빠른 변화에 대응 필요
    - 비즈니스 로직과 UI 로직이 서버에 종속
- 서버 애플리케이션 특징에 적합한 구조가 필요
    - 유연하게 대응할 수 있는 개발 효율성인 구조는 서버 애플리케이션의 수명을 유지

웹 애플리케이션 아키텍처

- 물리층인 티어(Tier)와 논리층인 레이어(Layer)로 구분



레이어	설명
프리젠테이션 층 (Presentation)	사용자 인터페이스와 컨트롤러를 제공 (Controller)
비즈니스 로직 층 (Business Logic)	비즈니스 로직을 제공 (Service)
데이터 엑세스 층 (Data Access)	데이터 엑세스를 추상화 (DAO: Data Access Object)

오목형 레이어(Layer)

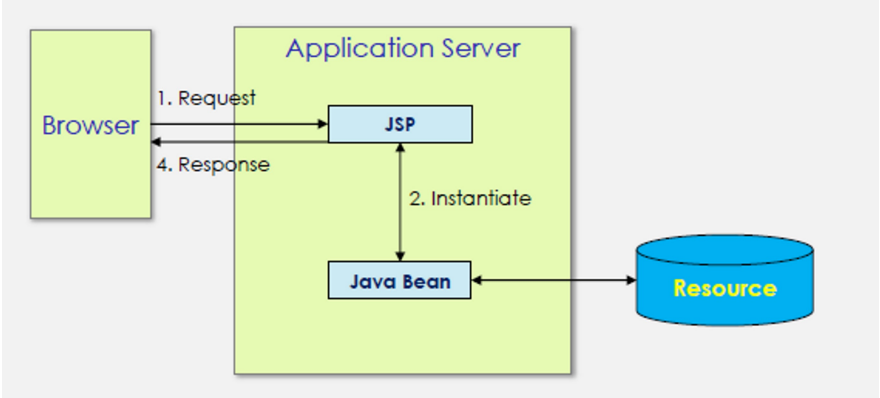
- 표현 층 또는 영속화 메커니즘이 변경되어도 비즈니스 로직층에는 영향을 최소화
- 레이어 층 도입과 함께 결합 부분에 약한 결합 설계 구현(인터페이스) 필요

프레젠테이션 층

- 사용자 인터페이스와 컨트롤러를 제공하는 역할
- 컨트롤러는 사용자 인터페이스를 통해 사용자의 입력을 받아 적정한 로직을 호출하고, 그 결과를 사용자 인터페이스로 변환하는 작업
- 다양한 사용자 인터페이스 등장 (AJAX 비동기 통신, 리액트, 앵귤러 클라이언트 프레임워크)

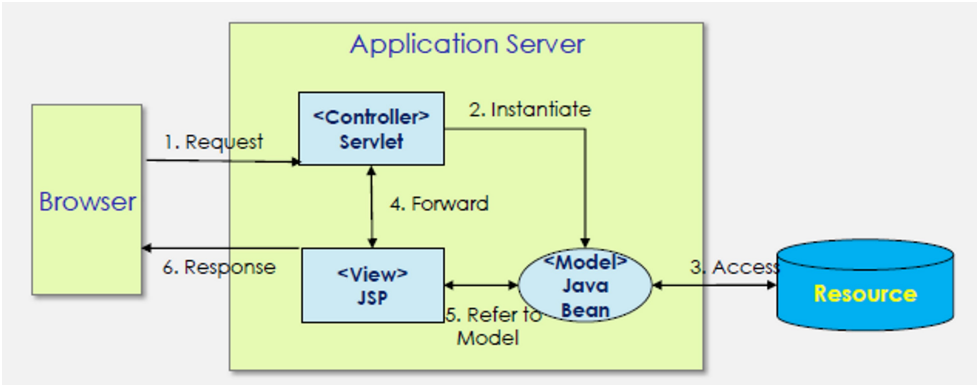
모델 1 방식

- JSP만 구현 개발하거나 Java bean을 포함하여 개발하는 방식을 의미
- JSP에 뷰와 비즈니스 로직이 혼재되어 복잡도가 높음 -> 유지보수 어려움



모델 2 방식(MVC 2) - 뷰와 비즈니스 로직의 분리로 유지 보수성 및 확장 용이

- 모델(Model): 뷰에 필요한 비즈니스 영역의 로직을 처리
- 뷰(View): 비즈니스 영역에 대한 프레젠테이션 뷰(결과 화면)을 담당
- 컨트롤러(Controller): 사용자의 입력 처리와 화면의 흐름 제어를 담당



비즈니스 로직

- 유즈케이스로 표현되는 특정 업무 처리를 위한 서비스와 도메인으로 구성
- 개발, 운영 업무의 경우 웹 앱의 기능 추가와 변경은 비즈니스 로직 변경

트랜잭션 스크립트 방식

- 비즈니스 로직(서비스 클래스에 포함)이 적은 일반적인 입출력 애플리케이션일 경우
- 도메인은 가능한 한 로직을 포함하지 않고 단순한 값만 저장
    - VO(Value Object)
    - DTO(Data Transfer Object)

도메인 모델

- 에릭 에반스가 제안한 도메인 주도 설계 개념
    - 업무 도메인 별로 분리하여 설계
    - 도메인 전문가가 도메인 모델 제공
- 최근 대규모 시스템의 복잡한 비즈니스 로직을 방지하기 위한 대안
- 도메인 패키지에 도메인 로직을 두는 방식

스프링 프레임워크는 도메인 모델의 구축과 별도의 큰 시너지 효과 없음

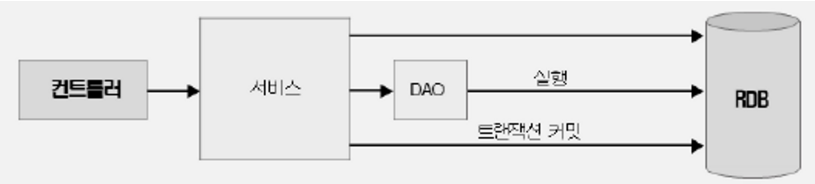
- 도메인의 생성과 관리는 DI컨테이너가 아닌 데이터 액세스 층의 구조에 의존

트랜잭션 관리

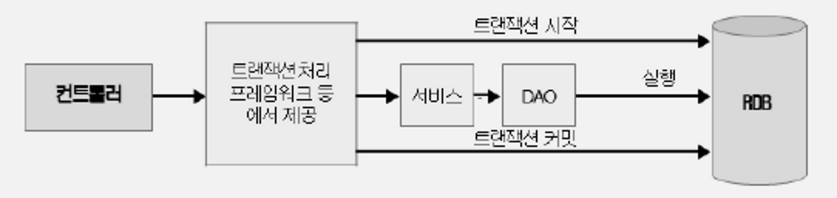
- 트랜잭션의 ACID 특성 중 원자성과 독립성을 관리
- 원자성: 트랜잭션 내의 모든 처리는 전부 실행됐거나 아무것도 실행되지 않음
- 독립성: 병행해서 실행되는 트랜잭션 간에는 간섭 받지 않고 독립적임

트랜잭션 경계
- 프리젠테이션과 비즈니스 로직 층 사이에 존재하는 것이 일반적임
- 프리젠테이션 층에 공개된 비즈니스 로직 층의 서비스 클래스가 트랜잭션 시작과 끝

명시적 트랜잭션
- 트랜잭션 시작과 커밋, 롤백과 같은 RDB에 대한 트랜잭션 관리를 소스 코드로 명시



선언적 트랜잭션
- 프레임워크에서 제공하는 정의 파일 선언을 통해 트랜잭션 관리



데이터 액세스 층
- RDB액세스를 비즈니스 로직으로 숨기고, 비즈니스 로직에 필요한 데이터를 테이블에서 취득해서 오브젝트에 매칭
- 이 때 오브젝트와 RDB를 매핑하는 것을 O/R매핑이라고 함

DB액세스 프레임워크의 종류
- Hybernate: 대표적인 ORM(O/R Mapping 프레임워크)
- Mybatis, 스프링, JDBC: SQL문 사용 전제로 한 DB 액세스 프레임워크

데이터 액세스 층의 설계지침
- 커넥션 풀 사용함
- RDB 제품 바껴도 구현 영향 X
- 이용하는 RDB에 의존적 SQL문 기술 X

부품화
- 개발 효율성과 유연성 향상을 위해서는 티어 또는 레이어를 통해 부품화
- 부품 큰 쪽: 티어, 레이어 // 작은 부품: 패키지, 컴포넌트
- 부품화를 위해서는 인터페이스가 중요

부품화를 위해서는 2가지 중요점
- 연결해야 할 부품 간에 결국 중요한 부품이 인터페이스를 가짐

- 절대적 기준은 없음

웹 앱 문제
- EJB의 문제(현재 해결됨)
- 오브젝트 생명주기
    - 서블릿 인스턴스의 호출로 인한 성능 저하 및 메모리 압박 - 싱글톤 구현 필요한데 복잡
    - HTTP Session/Request/Application경우 객체 생명주기 관리 필요
- 부품화 문제
    - 인터페이스 구현과 비의존 구현 위해서는 고도 기술 필요
    - 팩토리 메서드는 개발자가 new를 사용하지 않고 인스턴스화 구현
- 기술 은닉과 부적절한 기술 은닉
    - 여러 클래스에 걸쳐 존재하는 예외처리, 로깅, 트랜잭션은 프로그램 가독성 훼손, 부품화와 테스팅에 비효과적

>> 문제 해결은 스프링 프레임워크

웹 앱 문제를 스프링 프레임워크로 해결
- 오브젝트 생명주기 -> DI(Dipendency Injection)컨테이너로 해결
- 부품화 문제 -> DI컨테이너로 해결
- 기술은닉과 부적절한 기술 은닉 문제 AOP(Aspect Oriented Programming)로 해결

Spring Framework의 주요 특징
- POJO(Plain Old Java Object) 관리
    - **특정한 인터페이스를 구현**하거나 상속을 받을 필요가 없는 **가벼운 객체를 관리**
- IoC(Inversion of Control) 경량 컨테이너 (light-weight Container)
    - 필요에 따라 스프링 프레임워크에서 사용자의 코드를 호출함
    - **일반 오브젝트의 생애 주기 관리나 오브젝트 간의 의존관계를 해결하는 아키텍처를 구현**
- DIxAOP(Dependency Injection, Aspect Oriented Programming) 지원
    - DI: **각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결**시켜줌(변경: 용이성, 확장성, 품질관리 용이)
    - AOP: **트랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리**(프로그램 가독성, 기술 은닉)
- O/R 매핑 프레임워크 지원
    - 완성도가 높은 DB처리 라이브러리와 연결할 수 있는 인터페이스 제공(Mybatis, Hibernate)
- 스프링 데이터를 통한 다양한 데이터 연동 기능
    - 그래프DB, 다큐먼트 DB, NoSQL 등의 데이터 저장소를 액세스(JPA, MongoDB, Neo4j, Redis, Cassandra, Sola 등)
- 스프링 배치
    - 대량의 데이터를 일괄 처리, 병행 처리할 수 있는 템플릿 제공
- 스프링 시큐리티
    - 인증(Authentication), 인가(Authorization) 기능 제공
    - 베이직 인증, OAuth(Open Authorization) 개방형 표준 인증 서비스 제공