**프레임워크 프로그래밍**

**1주차 Report**

**그리기이(가) 표시된 사진

자동 생성된 설명**

**한국산업기술대학교**

**201715404컴퓨터공학과**

**권한길**

**Q1. 애플리케이션 아키텍처에서 애플리케이션 서버 티어를 구성하는 3가지 레이어에 대해 설명하고 결합도를 줄이기 위한 방법을 설명하시오.**

**A1.** 티어(Tier)에는 물리적으로 클라이언트, 중간층(애플리케이션), EIS(Enterprise Infomation System) 세가지로 나뉘고 , 다시 중간층을 논리적으로 세가지로 나눈 레이어(Layer)가 있다.

(웹 애플리케이션 서버 레이어 - 단방향 3개 층(Layer)으로 구성된다)

1. Presentation Layer 구성 : 컨트롤러(페이지 화면 전환 또는 동작 제어)

->사용자 인터페이스와 컨트롤러를 제공하는 역할을 합니다. 이 레이어에는 클래스 이름에 Controller 나 Action이 붙은 클래스가 배치됩니다.

2. Business Logic Layer 구성 : 서비스(유스케이스로 표현되는 특정 업무 처리, 트랜잭션 기점) , 도메인(서비스로부터 기능을 실행하는데 사용되는 고객 또는 주문 같은 클래스의 집합, 관련 정보들을 저장)

->비지니스 로직을 제공합니다. 이 층에는 Service가 붙은 유스케이스를 제어하는 클래스나 회사(Company)나 종업원(Employee), 주문(Order)등 업무 대상의 이름이 붙은 클래스가 배치됩니다.

3. Data Access Layer : DAO(Data Access object)

->데이터베이스 액세스를 추상화합니다. 이 층에는 클래스 이름 끝에 DAD(Data Access Object)가 붙은 클래스가 배치됩니다.

**결합도를 낮추는 방법 :**

(미리 알아야 할 정보)

* 결합도란 : 클래스간에 얼마나 연결이 되어있는지 나타내는것이다.
* 결합도가 높다면 : 결합도가 높은 프로젝트는 유지보수가 어렵고 시간이 많이 들기에 그만큼 비용소모도 커진다.
* 스프링의 사용 목적 : 결합도를 낮추기 위함.

1. **다형성**을 이용해서 결합도를 낮출 수 있다. (재사용성 증대)

2. **Factory 패턴** : Factory패턴은 클라이언트에서 사용할 객체 생성을 캡슐화하여 객체와 객체 사이의 느슨한 결합 상태로 만들어주므로 결합도를 낮춰줄 수 있다.

3. **AOP**(Aspect Oriented Programming): Spring의 핵심 개념중 하나인 DI가 애플리케이션 모듈들 간의 결합도를 낮춰준다면, AOP는 애플리케이션 전체에 걸쳐 사용되는 기능을 재사용하도록 지원한다.

4. 계층의 경계를 넘어갈 때는 반드시 특정 계층에 종속되지 않는 **오브젝트 형태**로 변환해줘야 한다. (계층사이의 낮은 결합도를 깨뜨리지 않도록 설계해야 한다.)

**Q2. 스프링 프레임워크의 특징 3가지 (POJO, loC, DlxAOP)를 설명하시오.**

**A2.**

1. POJO(Plain Old Java Object) 관리 : 특정한 인터페이스를 구현하거나 상속을 받을 필요가 없는 가벼운 객체를 관리하는 것

**POJO 특징**

* 특정 규약(contract)에 종속되지 않는다. (Java 언어와 꼭 필요한 API 외에 종속되지 않는다.)
* 특정 환경에 종속되지 않는다.
* 객체지향원리에 충실해야 한다.

**POJO를 사용하는 이유**

* 코드의 간결함 (비즈니스 로직과 특정 환경/low 레벨 종속적인 코드를 분리하므로 단순하다.)
* 자동화 테스트에 유리 (환경 종속적인 코드는 자동화 테스트가 어렵지만, POJO는 테스트가 매우 유연하다.
* 객체지향적 설계의 자유로운 사용

1. IoC(Inversion of Control) 경량 컨테이너(light-weight Container)

* 필요에 따라 스프링 프레임워크에서 사용자의 코드를 호출한다.
* 일반 오브젝트의 생애 주기 관리나 오브젝트 간의 의존관계를 해결하는 아키텍처를 구현
* 설명 : 스프링을 쓰기전에는 프로그램의 흐름을 제어하는 주체가 개발자였다면 스프링에서는 프로그램의 흐름을 프레임워크가 주도하게 된다. 객체의 생성부터 생명주기 관리를 컨테이너가 도맡아서 하게 된 것이다.

즉, 제어권이 컨테이너로 넘어가게 되고, 이것을 제어권의 흐름이 바뀌었다고 하여 IoC 라고 한다.

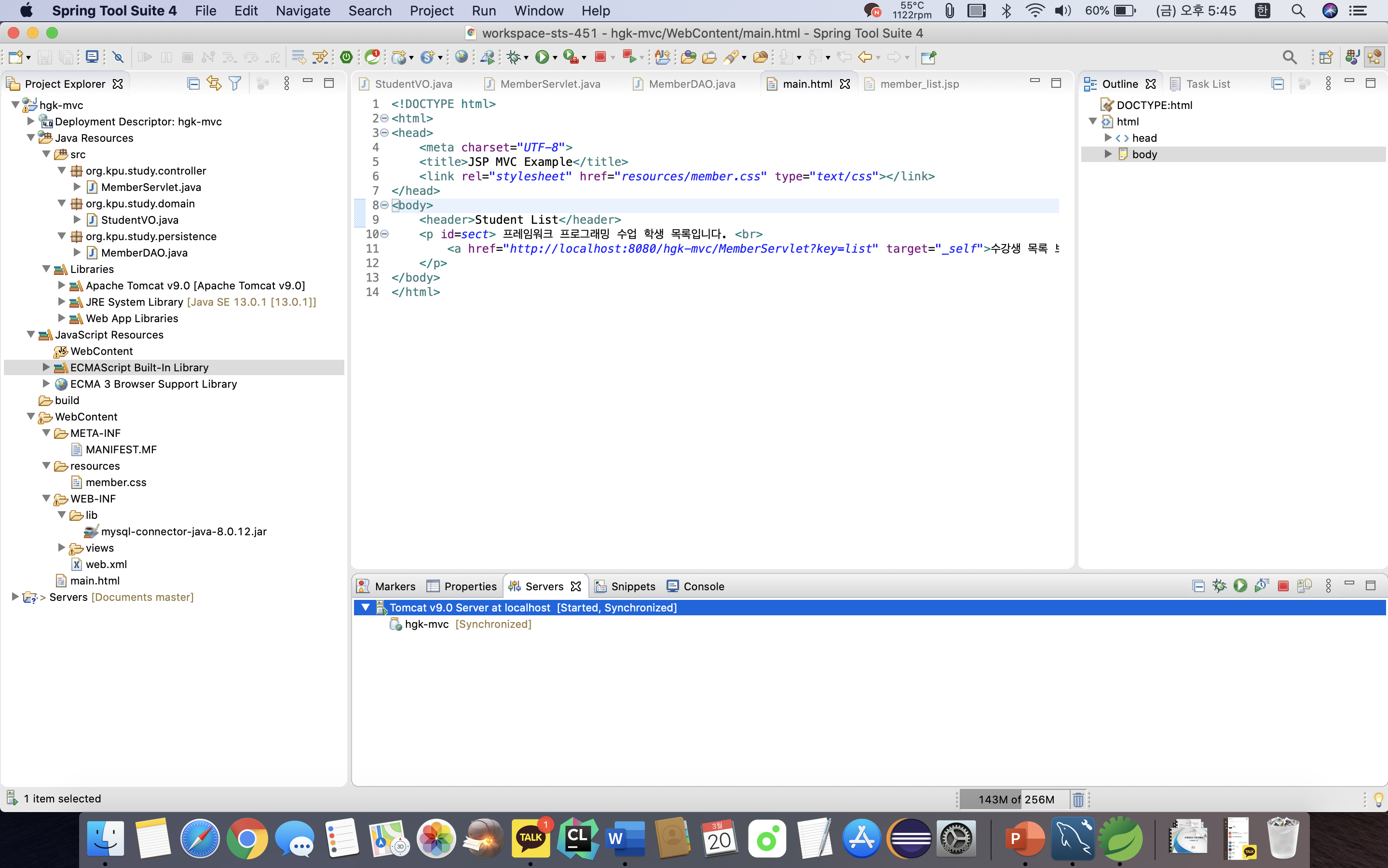
제어권이 컨테이너로 넘어옴으로써 DI(의존성 주입), AOP(관점 지향 프로그래밍) 등이 가능하게 된다.

1. DIxAOP(Dependency Injection, Aspect Oriented Programming) 지원

* **DI** : 각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결시켜준다. ( 변경 용이성, 확장성, 품질관리 용이)
* **AOP** : 트랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리 한다.( 프로그램 가독성, 기술 은닉)

**Q3. 모델 2방식의 첨부 예제를 참고하여 실습하고 결과 출력 페이지를 PrintScreen하여 첨부하시오.**

**<sts4 화면(main.html)>**



**<main.html 실행화면>스크린샷이(가) 표시된 사진

자동 생성된 설명**

**<수강생 목록 보기>**

스크린샷이(가) 표시된 사진

자동 생성된 설명