**1. 기반 기술 – JPA(Java Persistence API) 소개**

우리가 개발하여 생성하는 모든 DATA는 영속화 처리가 되어야 합니다. 하지만 객체 세계(OO World)와 관계형세계(Relational world)의 패러다임의 불일치로 개발자들은 영속화 처리를 위한 많은 비용을 지불하고 있습니다.

JAVA영역에서는 영속화 처리를 위해 퍼시스턴스 기술로서 자바 표준에 스펙을 정하고 그 구현을 각 DB벤더와 전문 업체에게 위임하는 JDBC에서 시작이 됩니다. 하지만 JDBC는 SQL코드와 결과에 대한 수많은 매핑 구문이 java Source에서 구현됨으로써 개발 및 유지보수가 어렵고 코드의 재사용성이 어렵다는 단점이 존재합니다.

이에 대한 대안으로 SQL Mapper인 MyBatis와 OR Mapper인 Hibernate, JPA와 같은 데이터 접근 프레임워크가 활용되고 있습니다.

JPA는 EJB3.0에서 독립되어 JAVA의 모든 기술과 환경에서 사용할 수 있는 독립적인 POJO기반의 퍼시스턴스 프레임워크로 관계형 데이터베이스에 접근하기 위한 표준 ORM기술을 제공합니다.  
즉 도메인 모델과 관계형 모델간의 매핑을 지원합니다.

JPA는 표준 인터페이스로 JDBC와 같이 별도의 구현체가 필요하며 현재 EclipseLink, OpenJPA , Hibernate등이 있으며 예제는 Hibernate를 구현체로 사용합니다.

**2. 구현 인터페이스**

예제에서는 도메인 모델에 선언된 아래의 I/F를 구현합니다.

~~~~~~~~(추가 필요)

**3. 내부 구조**

기반 기술에 독립적인 도메인 모델을 구현하기 위해 도메인 모델에는 JPA의 기술 SET이 적용될 수 없습니다. JPA는 도메인 객체에 Annotation선언을 통해 해당 객체의 속성을 정의 할 수 있습니다. 이에 따라 도메인 모델과 JPA 기술 Set이 선언된 객체간의 매핑이 필요합니다.

매핑 기술은 Dozer를 사용할지 생성자 패턴을 통해 매핑할지 고민 필요 ??????

**4. 고민해야 할 문제**