

24장. 스프링 의존성 주입과 제어 역전 기능



- ▶객체들간의 연관 관계를 개발자가 직접 코딩을 통해 컴포넌트(클래스)에 부여하는 것이 아니라 컨테이너가 연관 관계를 직접 규정하는 것
- ▶코드에서 직접적인 연관 관계가 발생하지 않으므로 각 클래스들의 변경이 자유로워짐 (loosely coupled, 약한 결합).

강한 결합과 약한 결합

프로그램은 각각의 독립적인 기능들로 구성되어 있음. 쇼핑몰의 경우 크게 상품 관리, 주문 관리, 회원 관리, 게시판 관리 등으로 구성됨. 각 기능들은 또 세부 기능을 하는 여러 클래스들로 이루어짐. 그런데 부품 기능을 하는 클래스에 변경 사항이 발생했을 때 그 클래스의 기능과 관련 없는 다른 클래스까지 손봐야 한다면 여러 가지 문제가 발생할 수 있음. 따라서 서로 관련이 있는 기능들은 강하게 결합(tightly coupled)하고, 관련이 없는 기능들은 약하게 결합(loosely coupled)해야 좋은 프로그램임, 그 반대가 되면 안 됨.

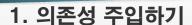


자바 코드로 구현 시 문제점

- 기존 게시판 BoardDAO 클래스에서는 오라클과 연동해 게시판 기능을 구현하고 있음
- 오라클에서 MySQL로 데이터베이스를 변경 발생 시 BoardDAO 클래스의 기능을 일일이 변경해야함
- 더 나아가서 BoardDAO 클래스를 사용하는 BoardService 클래스의 기능도 변경해야 할 수도 있음



- 자바 코드에서 직접 객체를 생성해서 사용하는 것(tightly coupled)은 복잡한 문제를 일으킬 수 있음
- 다른 클래스의 변경 사항이 연속적으로 다른 부분에 영향을 미친다면 이 방법(자바 코드에서 직접 객체를 생성해 사용하는 것)은 좋은 방법이 아님



의존성 주입 장점

- 클래스들 간의 의존 관계를 최소화하여 코드를 단순화할 수 있음
- •애플리케이션을 더 쉽게 유지 및 관리할 수 있음
- 기존 구현 방법은 개발자가 직접 코드 안에서 객체의 생성과 소멸을 제어했지만 의존성 주입은 객체의 생성, 소멸과 객체 간의 의존 관계를 컨테이너가 제어함

제어의 역전(Inversion Of Control)

- •기존 코드에서는 개발자가 직접 객체를 제어했지만 스프링 프레임워크에서는 객체의 제어를 스프링이 직접 담당
- IoC의 종류도 여러 가지이며, 일반적으로 스프링에서는 DI로 IoC의 기능을 구현하므로 IoC보다는 DI라는 용어를 더 많이 사용함



감사합니다.