Chapter 1. 스프링 속으로

주요 개념

* Spring에서의 Bean : 애플리케이션 컴포넌트를 통틀어 이르는 말
* POJO(Plain-Old Java Object)
* 평범한 자바 객체
* EJB에서 bean은 특정 인터페이스 구현 혹은 클래스를 확장을 강요하지만 Spring에서 POJO는 어떤 강요도 받지 않는(=비침투적인, non-invasive) bean이 될 수 있다.
* DI(Dependancy Injection)
* 객체가 필요로 하는 객체를 생성하지 않고 외부에서 주입 받는다.
* 메소드 내에서 직접 객체를 생성하는 것은 강한 결합을 유도한다. 요구사항이 바뀌어 다른 클래스로 객체를 생성해야한다면 이 객체에 맞게 잦은 코드 수정을 요구하기 때문이다.
* DI의 핵심은 인터페이스다. 객체가 필요로 하는 객체에 대한 공통된 인터페이스를 알고 있다면 DI가 가능하다. 인터페이스를 통해 객체를 주입받는 것이 곧 DI이다.
* DI를 통해 클래스간의 결합도를 줄일 수 있다.
* AOP
* Aspect Oriented Programming : 관점지향 프로그래밍
* 관심사들을 서로 분리한다.
* 시스템 설계를 하다보면 컴포넌트 본연의 기능 외에 로깅, 트랜잭션 등의 횡단관심사(cross-cutting concerns, 여러 시스템에 걸쳐 관련된 혹은 공통된 기능)가 나타나는 경우가 많은데 이를 따로 분리하는 기술을 말한다.
* 각 컴포넌트에서 횡단관심사를 모두 구현하는 경우 공통된 코드가 산재되어 있게 되고, 코드가 지저분해져 유지보수에 어려움을 겪게 되기때문에 분리가 필요하다.
* AOP는 시스템의 각 컴포넌트 서비스를 모듈화해서 선언적으로 활용할 수 있도록 하며, 각 컴포넌트가 자신의 책임에 집중할 수 있도록 하게 만든다.
* Template
* JDBC와 같은 상투적인 코드를 캡슐화하여 편리한 사용과 반복적인 코드를 제거한다.
* Container
* Spring의 객체의 생명주기를 관리
* 크게 두 가지로 분류할 수 있는데, BeanFactory, ApplicationContext가 이에 해당한다.
* BeanFactory는 DI를 제공하는 가장 단순한 컨테이너로 지나치게 저수준의 기능을 제공하므로 사용이 용이하지 않을 때가 있다.
* ApplicationContext는 BeanFactory를 확장하여 주로 이를 통해 Spring의 빈들을 활용한다. ApplicationContext는 대표적으로 다음과 같은 것들이 있다.

> ClassPathXmlApplicationContext : 클래스패스에 위치한 xml 로드

> FileSystemXmlApplicationContext : 파일 경로상에 위치한 xml 로드

> XmlWepApplicationContext : 웹 어플리케이션에 포함된 XML 로드

빈의 생명주기

* 스프링이 빈을 인스턴스화한다.
* 각종 필드값은 빈에 주입한다.
* BeanNameAware 구현시 setBeanFactory 메소드를 통해 bean id를 넘긴다.
* BeanFactorAware시 setBeanFactory를 호출해 빈 팩토리를 넘긴다.
* ApplicationContextAware를 구현하면 setApplicationContext를 호출하고 컨테이너(여기선 Application Context)를 넘긴다.
* BeanPostProcessor를 구현하면 postProcessBeforeInitialization을 호출
* InitializingBean 구현시 postProcessAfterInitialization을 호출
* BenbPostProcessor 구현시 postProcessAfterInitialization을 호출, Bean 사용 준비 완료
* DisposableBean 인터페이스 구현시 destroy 메소드 호출

스프링 모듈의 구성

* 코어 스프링 컨테이너 : 빈 생성, 설정, 처리를 관리하는 컨테이너
* AOP 모듈
* 데이터 액세스 모듈(JDBC, Transaction, ORM 등)
* 웹, 리모팅(Web, Survlet, Struts 등)
* 도구(Tomcat 등)
* 테스팅 : 단위, 통합 테스트를 위한 모듈

끝