**DI(Dependency Injection) 개념 및 활용**

// 파싱 : 해석해서 객체를 만드는 것

// xml을 파싱해서 bean 을 만든것이 ioc 컨테이너

스프링 컨테이너에 파싱된 빈이 만들어져 있는 것

Spring에서는 Spring Container, IoC Container라는 개념을 사용한다.

**Container는 인스턴스의 생명주기를 관리**하며, **생성된 인스턴스들에게 추가적인 기능을 제공하도록 하는 것**이다.

즉 **Container는 개발자가 작성한 코드의 처리과정을 위임받은 독립적인 존재**라고 생각하면 된다.

**Container**는**적절한 설정**만 되어있다면 누구의 도움 없이도 **작성한 코드를 스스로 참조한 뒤 알아서 객체의 생성과 소멸을 컨트롤**해준다.

**Spring Container는 Spring Framework의 핵심부에 위치**하며, **종속 객체 주입을 이용하여 Application을 구성하는 Component들을 관리**한다.

이때 **Spring Container에서 생성되는 객체를 Bean**이라고 한다.

Bean은 **Spring IoC Container가 관리**하는 자바 객체, **Spring Bean Container에 존재하는 객체**를 말한다.

***Spring IoC(Inversion of Control) Contatiner***에 의해 ***인스턴스화, 관리, 생성***된다.

**Bean Container는 의존성 주입을 통해 Bean 객체를 사용할 수 있도록 해준다.**

**1.객체(클래스 생성) -> 2.bean 생성 -> 3.AbstractApplicationContext 으로 bean 가져오기**

 - 스프링은 객체를 생성하고 각각의 객체를 연결해주는 조립기 역할을 함.  
 - 여기에 있는 GenericXmlApplicationContext 객체가 조립기 기능을 구현한 클래스.  
 - 조립기에서 생성할 객체가 무엇이고, 각 객체를 어떻게 연결하는 지에 대한 정보는 xml 파일에 설정이 되어 있음.  
 - GenericXmlApplicationContext 클래스는 xml 파일에 정의된 설정 정보를 읽어와서 객체를 생성하고,  
   각각의 객체를 연결한 뒤에 내부적으로 보관함.  
 - xml을 이용한 스프링 설정 파일에서는 컨테이너가 생성할 객체를 지정하기 위해 <bean> 태그를 사용함.  
 - 스프링 컨테이너가 생성해서 보관하는 객체를 스트링 빈(bean) 객체라고 부르며 일반적으로 자바 객체와 동일함.  
 - 스프링 컨테이너는 생성한 빈 객체를 <이름, 빈 객체> 쌍으로 보관함.  
 - 스프링 컨테이너가 보관하고 있는 객체를 사용하고 싶은 경우 빈 객체와 연결되어 있는 이름을 이용해서 객체 참조.

 - AbstractApplicationContext : 컨테이너 종료(close)와 같은 기능을 제공해 주는 객체.  
 - GenericXmlApplicationContext : AbstractApplicationContext 객체를 상속 받아 만드는 클래스. xml 파일에서 스프링 빈 설정 정보를 읽어오는 역할.  
 - GenericXmlApplicationContext 객체를 생성할 때 파라미터 값으로 xml의 경로를 전달하여 설정 파일로 사용함.  
 - GenericXmlApplicationContext 객체는 전달받은 xml 파일에서 설정 정보를 읽어와서 스프링 빈 객체를 생성하고 프로퍼티 값을 설정함.  
   생성된 스프링 빈 객체는 getBean() 이라는 메서드를 사용해서 구현함.  
   getBean() 메서드의 첫번쨰 파라미터는 구현하고자 하는 스프링 빈 객체의 고유한 id이름이며, 두번째 파라미터는 그 객체의 타입을 의미함.