hwsf01_대전_05반_정세원

Spring DI (Dependency Injection)

IoC(Inversion of Control - 제어의 반전) 유형

1. Dependency Lookup
    - 컨테이너가 looup context를 통해서 필요한 Resource나 Object를 얻는 방식
    - Lookup한 Object를 필요한 타입으로 Casting 해 주어야 함.
2. Dependency Injection
    - Object에 lookup 코드를 사용하지 않고 컨테이너가 직접 의존 구조를 Object에 설정할 수 있도록 지정해주는 방식
    - Object가 컨테이너의 존재 여부를 알 필요X
    - Setter Injection, Constructor Injection, Method Injection

Container

- 객체의 생성, 사용, 소멸에 해당하는 라이프 사이클을 담당
- 라이프 사이클을 기본으로 애플리케이션 사용에 필요한 주요 기능을 제공
- 기능 : 라이프 사이클 관리, Dependency 객체 제공, Thread 관리, 기타 애플리케이션 실행에 필요한 환경
- Container의 필요성

  : 비즈니스 로직 외에 부가적인 기능들에 대해서는 독립적으로 관리되도록 하기 위함

  : 서비스 lookup 이나 Configuration에 대한 일관성을 갖기 위함

  : 서비스 객체를 사용하기 위해 각각 Factory 또는 Singleton 패턴을 직접 구현하지 않아도 됨
- Ioc Container

  : 오브젝트의 생성과 관계 설정, 사용, 제거 등의 작업을 애플리케이션 코드 대신 독립된 컨테이너가 담당

  : 코드 대신 오브젝트에 대한 제어권을 갖고있음

  : 스프링에서 IoC를 담당하는 컨테이너에는 BeanFactory, ApplicationContext가 있음

Spring DI Container

1. 빈(Bean)
    - 스프링이 IoC 방식으로 관리하는 오브젝트
    - 스프링이 직접 그 생성과 제어를 담당하는 오브젝트

2. 빈 팩토리(BeanFactory)
    - 스프링이 IoC를 담당하는 핵심 컨테이너
    - Bean을 등록, 생성, 조회, 반환하는 기능을 담당
    - 일반적으로 BeanFactory를 바로 사용하지 않고 이를 확장한 ApplicationContext를 사용함.
3. 애플리케이션 컨텍스트(ApplicationContext)
    - BeanFactory를 확장한 IoC 컨테이너
    - Bean을 등록하고 관리하는 기본적인 기능은 BeanFactory와 동일
    - 스프링이 제공하는 각종 부가 서비스를 추가로 제공
    - BeanFactory라고 부를 때는 주로 빈의 생성과 제어의 관점에서 이야기하는 것이고, 애플리케이션 컨텍스트라고 할 때는 스프링이 제공하는 애플리케이션 지원 기능을 모두 포함해서 이야기하는 것
4. 스프링 프레임워크
    - IoC 컨테이너, ApplicationContext를 포함해서 스프링이 제공하는 모든 기능을 통틀어 말할 때 주로 사용
5. 싱글톤 빈(Singleton Bean)
    - 스프링 빈은 기본적으로 싱글톤으로 만들어짐
    - 컨테이너가 제공하는 모든 빈의 인스턴스는 항상 동일
    - 컨테이너가 항상 새로운 인스턴스를 반환하게 만들고 싶을 경우 scope를 prototype으로 설정해야 함
      생성 범위 : singleton, prototype, request, session

Annotation

빈으로 사용될 클래스에 특별한 annotation을 부여하여 자동으로 빈 등록

- 빈 자동등록에 사용할 수 있는 annotation
- @Autowired : Spring Framework에서 지원하는 Dependency 정의 용도의 Annotation으로, Spring Framework에 종속적이긴 하지만 정밀한 Dependency Injection이 필요한 경우에 유용
- Annotation으로 빈을 설정할 경우 반드시 component-scan을 설정해야 함
- @Repository, @Service, @Controller, Component

```
@Component("memberService") //annotation을 이용하여 빈 등록
@Scope("singleton") //annotation을 이용한 scope 설정
public class MemberServiceImpl implements MemberService{
  @Override
  public int registerMember(MemberDto memberDto){
    return 0;
  }
}
```

```
<bean id="memberDao" class="com.test.hello.dao.MemberDaoImpl"/>
<bean id="memberService" class="com.test.hello.service2.MemberServiceImpl" scope="prototype">
//xml 파일을 이용한 scope 설정
  <property name="memberDao" ref="memberDao"/>
</bean>
//bean 태그를 이용하여 세밀한 제어 가능
```

DI(Dependency Injection)

- 객체 간의 의존 관계를 자신이 아닌 외부의 조립기가 수행 → Spring Container가 DI 조립기 제공
  : 스프링 설정 파일을 통하여 객체 간의 의존관계를 설정
  : Spring Container가 제공하는 API를 이용해 객체를 사용
- 제어의 영행(inversion of Control, IoC)라는 의미로 사용
- DI를 통해 시스템에 있는 각 객체를 조정하는 외부 객체가 객체들에게 생성 시에 의존관계를 주어짐
- 느슨한 결합(loose coupling)의 주요 강점 : 객체는 인터페이스에 의한 의존 관계만을 알고 있으며, 이 의존 관계는 구현 클래스에 대한 차이를 모르는 채 서로 다른 구현으로 대체 가능

XML Spring 설정

기본설정 - 빈 객체 생성 및 주입

1. 주입할 객체를 설정 파일에 설정
   - <bean> : 스프링 컨테이너가 관리할 Bean 객체를 설정
2. 기본 속성
   - name : 주입받을 곳에서 호출 할 이름 설정
   - id : 주입 받을 곳에서 호출할 유일한 이름 설정
   - class : 주입할 객체의 클래스
   - factory-method : singleton 패턴으로 작성된 객체의 factory 메소드 호출

```
<bean id="memberDao" class="com.test.hello.dao.MemberDaoImpl"/>

<bean id="memberService" class="com.test.hello.service2.MemberServiceImpl" scope="prototype">
  <property name="memberDao" ref="memberDao"/>
</bean>

<bean id="adminService" class="com.test.hello.service2.AdminServiceImpl"/>
```

3. 빈 객체 얻기
   - 설정 파일에 설정한 bean을 Contatiner가 제공하는 주입기 역할의 api를 통해 주입 받는다.

```
ApplicationContext context =
  new ClassPathXmlApplicationContext("com/text/hello/controller4/applicationContext.xml");
```

```
CommonService kor1 = context.getBean("MemberService", MemberService.class);
CommonService kor2 = context.getBean("adminService", adminService.class);
```

Constructor

1. 객체 또는 값을 생성자를 통해 주입 받는다.
2. <constructor-arg> : <bean>의 하위태그로 설정한 bean 객체 또는 값을 생성자를 통해 주입하도록 설정
   - 설정 방법 : <ref>,<value>와 같은 하위태그를 이용하여 설정하거나 또는 속성을 이용하여 설정
     1. 하위태그 이용

        a. 객체 주입 시 : <ref bean="bean name"/>
        b. 문자열(String),primitive date 주입 시 : <value>값</value>
     2. 속성 이용
        a. 객체 주입 시 : <constructor-arg ref="bean name"/>
        b. 문자열(String),primitive date 주입 시 : <constructor-arg value="값"/>

Property

1. property를 통해 객체 또는 값을 주입받는다.
   → setter method (하나의 값만 받을 수 있음)
2. <property> : <bean>의 하위태그로 설정한 bean 객체 또는 값을 property를 통해 주입하도록 설정
   - 설정 방법 : <ref>,<value>와 같은 하위태그를 이용하여 설정하거나 또는 속성을 이용하여 설정
     1. 하위태그 이용

        a. 객체 주입 시 : <ref bean="bean name"/>
        b. 문자열(String),primitive date 주입 시 : <value>"값"</value>
     2. 속성 이용 : name - 값을 주일할 propoerty 이름(sertter의 이름)
        a. 객체 주입 시 : <property name ="propertyname" ref="bean name"/>
        b. 문자열(String),primitive date 주입 시 : <property name ="propertyname" value="값"/>
     3. xml namespace를 이용하여 설정