KH정보교육원

Spring DI 관리

XML 단독 사용

모든 Bean을 명시적으로 XML에 등록하는 방법

장점	- 생성되는 모든 Bean을 XML에서 확인할 수 있다 - 프로젝트를 운영하는 입장에서 관리의 편의성이 높다.
단점	- Bean의 개수가 많아지면 XML 파일을 관리하기 어렵다 - 여러 개발자가 같은 설정 파일을 수정하면 설정에 충돌이 발생할 수 있다 - DI에 필요한 적절한 setter 메소드 또는 생성자가 코드 내부에 반드시 존재해야 한다

XML과 빈 스캐닝 (Bean Scanning)의 혼용

Bean으로 사용될 클래스에 특별한 어노테이션(Annotation)을 부여하고 Spring 컨테이너가 이를 통해 자동으로 Bean을 등록하며, 이 방식을 빈 스캐닝(Bean Scanning)을 통한 자동인식 Bean 등록기능이라고 한다.

장점	- 어노테이션을 부여하고 자동 스캔으로 빈을 등록하면 XML 문서 생성과 관리에 따른 수고를 덜어주고 개발 속도를 향상시킬 수 있다 - 개발자 간 XML 설정 파일의 충돌을 최소화 할 수 있다
단점	- 애플리케이션에 등록될 Bean이 어떤 것들이 있고, Bean들 간의 의존관계가 어떻게 되는지를 한 눈에 파악할 수 없다

KH정보교육원

Spring DI Annotation

Spring DI Annotation

KH정보교육원

@Annotation

xml파일에는 구동시킬 필수요소만 작성하고 소스코드에 Annotaion으로 표시하여 구동하는 방식

xml 파일

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd">
    <!-- 어노테이션 적용 -->
    <context:component-scan base-package="com.kh.mvc2"></context:component-scan>
```

소스코드

```java
package com.kh.mvc2.board.controller;

import java.util.List;

@Controller
public class BoardController{
    @RequestMapping(value="/board.do", method=RequestMethod.GET)
    public ModelAndView getListGet(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("get");
        BoardVo board=new BoardVo();
        List<BoardVo> list=new BoardDAO().getList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("boards",list);
        mv.setViewName("/WEB-INF/board/boardlist.jsp");

        return mv;
    }
    @RequestMapping(value="/board.do", method=RequestMethod.POST)
    public ModelAndView getListPost(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("Post");
        BoardVo board=new BoardVo();
        List<BoardVo> list=new BoardDAO().getList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("boards",list);
        mv.setViewName("/WEB-INF/board/boardlist.jsp");

        return mv;
    }
```

Spring DI Annotation

KH정보교육원

Bean 등록 Annotation

@Component	객체(컴포넌트)를 나타내는 일반적인 타입으로 <bean> 태그와 동일한 역할
@Repository	퍼시스턴스(persistence) 레이어, 영속성을 가지는 속성(파일, 데이터베이스)을 가진 클래스 EX) Data Access Object Class
@Service	서비스 레이어, 비즈니스 로직을 가진 클래스 EX) Service Class
@Controller	프리젠테이션 레이어, 웹 어플리케이션에서 View에서 전달된 웹 요청과 응답을 처리하는 클래스 EX) Controller Class

※ @Repository, @Service, @Controller는 특정한 객체의 역할에 대한 @Component의 구체화된 형태이다.

Spring DI Annotation

KH정보교육원

Bean 의존관계 주입 Annotation

@Autowired 와 @Resource 어노테이션은 @Component 로 의존관계를 설정한 객체로부터 의존 관계를 자동으로 주입해주는 어노테이션이다.

@Autowired는 타입으로, @Resource는 이름으로 연결해주는 점이 다르다.

Spring DI Annotation

KH정보교육원

@Autowired

◎ 정밀한 의존관계 주입 (Dependency Injection)이 필요한 경우에 유용하다.

◎ @Autowired는 필드 변수, setter 메소드, 생성자, 일반메소드에 적용 가능하다.

◎ 의존하는 객체를 주입할 때 주로 Type을 이용하게 된다.

◎ @Autowired는 <property>, <constructor-arg> 태그와 동일한 역할을 한다.

※ **@Qualifier** : @Autowired 와 함께 쓰이며, 한 프로젝트 내에 @Autowired로 의존성을 주입하고자 하는 객체가 여러 개 있을 경우, @Qualifier("name")를 통해 원하는 객체를 지정하여 주입할 수 있다.

Spring DI Annotation

KH정보교육원

@Resource

◎ 어플리케이션에서 필요로 하는 자원을 자동 연결할 때 사용된다.

◎ @Resource는 프로퍼티, setter 메서드에 적용 가능하다.

◎ 의존하는 객체를 주입할 때 주로 Name을 이용하게 된다.

@Value

◎ 단순한 값을 주입할 때 사용되는 어노테이션이다.

◎ @Value("Spring")은 <property .. value="Spring" /> 와 동일한 역할을 한다.

Spring DI Annotation

KH정보교육원

<context:component-scan> 태그

@Component를 통해 자동으로 Bean을 등록하고, @Autowired로 의존관계를 주입받는 어노테이션을 클래스에서 선언하여 사용했을 경우에는 해당 클래스가 위치한 특정 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.

<context:component-scan> 태그 예시

```
<context:component-scan base-package="com.kh.firstSpring" />
```

※ <context:include-filter>태그와 <context:exclude-filter>태그를 같이 사용하면 자동 스캔 대상에 포함시킬 클래스와 포함시키지 않을 클래스를 구체적으로 명시할 수 있다.