**2023-2024学年第2学期**

**J2EE高级框架作业**

**20232160A0934张嘉辉**

**1.谈谈你对I0C原理的理解(从历史发展、API和容器角度述说)**

历史角度：IoC 最早由 Martin Fowler 在 2004 年提出，但在此之前，类似的概念已经存在 ，比如：1988年，Ralph E. Johnson & Brian Foote 在《Designing Reusable Classes》中就提出“Inversion of control”（控制反转）

API角度：IoC 是一种设计原则，它将组件的控制权从应用程序转移到了 IoC 容器。同时IoC 容器 负责创建、配置和管理组件，应用程序只需使用这些已经准备好的组件。在 Java 中，我们通过 setter 方法注入或 构造函数注入来实现 IoC。

容器角度：IoC 容器是 IoC 的核心，它负责组件的生命周期和依赖关系。

**2．Spring框架中Bean管理有哪几种方式，分别是什么，简要描述一下**

XML文件：这是最传统的方式，通过在 XML 配置文件中定义 Bean 的信息。使用 <bean> 标签来指定 Bean 的类名、属性、依赖关系等。

基于注解：使用注解来标记 Bean 类和属性。常用的注解包括 @Component、@Service、@Repository、@Controller 等

基于Java：使用 Java 类来配置 Bean。常用的配置类包括 @Configuration 和 @Bean

**2024.3.19**

**1. 属性注入有哪几种方式？**

Setter注入：bean的setter方法注入属性值；构造器注入：bean的构造方法注入属性值；字段注入：使用注解直接访问类的字段来注入属性值。

**2.假设有两个对象A和B，如果A对象调用B对象并调用B对象的方法，使用配置文件的方式如何实现？请说一下实现思路。**

在Spring配置文件中定义B对象的bean;在A对象的bean定义中，使用<property>或<constructor-arg>标签来引用B对象的bean;确保A对象在调用B对象的方法前，B对象已经被正确初始化并注入到A对象中。

**3.简要说明一下使用注解开发的过程和主要的注解，并说明下注解的作用。**

引入Spring相关的注解库；在配置文件中开启注解扫描（使用<context:component-scan>标签）；在需要Spring管理的类上使用相应的注解（如@Component, @Service, @Repository, @Controller）；对于依赖注入，使用@Autowired或@Resource注解；使用@Value注解注入简单值或配置文件中的属性值；使用@PostConstruct和@PreDestroy注解标记初始化和销毁方法。

**4.假设有两个对象A和B，如果A对象需要调用B对象并调用B对象的方法，使用注解的方式如何实现？请说下实现思路。**

在A类中，为B类型的属性添加@Autowired注解，以自动装配B对象;确保B对象是一个Spring管理的bean（例如，使用@Component注解）;启动Spring容器时，Spring会自动检测@Autowired注解，并将匹配的bean注入到A类中;A类可以直接调用B对象的方法。

**5.在配置文件和注解混合开发中，如果Bean对象的创建交给配置文件，引用类型的注入交给注解，如何实现，测试类又是如何实现测试？**

在配置文件中定义需要手动创建的Bean对象;在需要自动注入的类上使用@Autowired注解，并确保这些类在Spring的组件扫描路径中;在测试类中，可以使用Spring的测试框架（如@RunWith(SpringJUnit4ClassRunner.class)）来加载Spring配置;在测试类中注入需要测试的Bean对象，并编写测试用例来测试对象之间的交互和方法的调用。

测试类实现测试：使用Spring的测试注解（如@ContextConfiguration）来指定Spring配置文件的位置；使用@Autowired注解来注入需要测试的Bean对象；编写测试用例，调用Bean对象的方法，并验证方法调用的结果是否符合预期；使用断言库（如JUnit的Assert类）来编写断言语句，验证方法的返回值或状态是否满足预期条件。

**2024.3.26**

**1.AOP是什么，解决了什么问题，实现原理，应用场景：**

AOP是一种编程范式，旨在通过预编译方式和运行期动态代理实现程序功能的统一维护。AOP将那些与业务逻辑无关，却为业务模块所共同调用的逻辑或责任（例如日志记录，事务管理，权限控制等）封装起来，便于减少系统的重复代码，降低模块之间的耦合度，并有利于未来的可维护性和可扩展性。

实现原理：AOP主要通过动态代理机制将切面织入到目标对象的方法中。在运行时，AOP框架会根据切点定义拦截目标方法，并在目标方法执行前后插入切面逻辑。

应用场景：AOP在软件开发中有广泛的应用，如日志记录、安全控制、性能监控、缓存管理、事务管理以及异常处理等。

**2.AOP编程基本步骤及基本实现：**

定义切面：确定需要增强的功能，并创建相应的切面类。

定义切点：确定切面的织入点，即哪些方法需要被增强。

定义通知：规定AOP执行的时机和执行的方法，如前置通知、后置通知、异常通知等。

配置AOP：将切面、切点和通知配置到Spring容器中，使Spring能够自动识别和应用AOP。

**3.Spring中AOP的配置方式：**

基于XML配置：在Spring的XML配置文件中定义切面、切点和通知，并通过相关元素将它们组合在一起。

基于注解配置：使用Spring提供的注解（如@Aspect、@Pointcut、@Before、@After等）在Java代码中定义切面、切点和通知。

**4.Spring中AOP的通知有哪些基本类型：**

前置通知（Before advice）：在目标方法执行之前执行的通知。

后置通知（After returning advice）：在目标方法成功执行之后执行的通知。

异常通知（After throwing advice）：在目标方法抛出异常时执行的通知。

最终通知（After (finally) advice）：无论目标方法正常返回还是抛出异常，该通知都会执行。

环绕通知（Around advice）：这是最强大的通知类型，可以在目标方法调用前后插入自定义逻辑，甚至可以控制是否调用目标方法。

**5．Spring中AOP是如何为Bean对象创建代理对象的：**

Spring AOP通过JDK动态代理或CGLIB代理为Bean对象创建代理对象。对于接口代理，Spring使用JDK动态代理；对于类代理，Spring使用CGLIB库来创建子类代理。

在代理对象创建过程中，Spring会检查目标对象实现的接口或继承的父类，并基于这些信息生成代理类的字节码。然后，通过反射机制加载并实例化代理类，使其能够拦截目标方法的调用，并在调用前后插入切面逻辑。

**6.Spring中AOP切面的执行顺序如何指定：**

在Spring AOP中，可以通过在切面类上使用@Order注解或在配置文件中指定顺序来定义切面的执行顺序。默认情况下，切面按照它们被定义在配置文件中的顺序执行。

如果定义了多个切面类，并且里面有相同的处理环节（如前置通知、后置通知等），那么需要手动指定处理顺序。可以使用@Order注解并指定一个整数值，值越小优先级越高，即先执行。另外，也可以在配置文件中通过配置元素的顺序来指定执行顺序。