核心功能有IOC DI AOP 事务 包括bean可能也会问

IOC 控制反转相当于把对象的控制权交给spring来处理 IOC就相当于一个MAP用来处理对象 有点类似一个超级工厂 程序员只需要配置好文件 不用理睬对象是怎么创建的 底层是什么 直接应用就可以了 增加了可维护性 和降低开发难度

早期是通过xml文件配置bean的 后来太麻烦 改成注解形式配置 就变成了 springboot

IOC初始化过程

XML读取Resource然后解析Bean 最后在BeanFactory里面注册

AOP 切面编程

能够将那些业务无关 却可以为业务模块一起共同调用和封装 比如事务处理 日志管理 便于减少系统的重复 降低模块的耦合度

AOP是基于动态代理的 有两种 实现接口的用JDK动态代理 没实现的用CGlib动态代理来处理也可以使用ASpectJ（最完整的生态框架）。

目的 把一些通用的功能抽象出来，在需要用到的地方直接使用即可，增加新功能也方便，比如我要在多个模块里面增加测试功能，测试完后就不要 总不能一个个模块添加删除吧 这时候AOP很好解决了此类问题的扩展性

SpringAOP和AspectJ `AOP区别

SpringAOP属于运行时增强 AspectJ是编译时增强 SpringAOP是基于代理 AspectJ基于字节码操作

SpringAOP集成了AspectJ 切面比较少的情况下用SpringAOP 多的话用后者

AOP例子

上述测试 还有要为每个流程都切入一个用户登录 此时 AOP也可以用

共有4个增强类型

前置Before Advice After Advice Around Advice Throws Advice 后置 环绕和异常

Bean的五个作用域

Singleton 唯一的bean实例 单例

Protype 每次请求都会创建一个新的bean

Request 每一次http请求都会产生一个bean 当前request内有效

Session 每一次http 都会产生一个新的bean 当前 session内有效

Global-session 全局session作用域 仅仅在基于Portlet的web应用中才有意义

Bean的多线程是不安全的

解决方式 定义一个Threadlocal 成员变量 将需要的可变成员变量保存在threadlocal 中

Bean的生命周期分为四个阶段和多个扩展点

1. 实例化 Instantiation
2. 属性赋值 Populate
3. 初始化 Initialization
4. 销毁 Destruction

主要是前三个，集中在了doCreatBean这个类里面

扩展点

影响多个bean的两父子---AOP实现的原理

说到AOP和自动注入 有关的spring扩展涉及两个接口 instantiationAwareBeanPostProcess

BeanPostProcess 一个在实例化前后调用 一个在初始化前后调用

BeanPostProcess是父类 另一个是子类

两父子功能丰富 常用于用户自定义扩展

影响单个bean的接口

Aware接口

Aware接口的作用就是让我们能够拿到spring容器中的一些资源

是在初始化阶段之前调用的

Aware也分为很多接口 基本分为两组 第一组先执行 第二组后执行

分组是有规律的

ApplicationContext和BeanFactory的区别

BeanFactory Spring里面最底层的接口 提供了最简单的容器功能 只提供了实例化对象和拿的对象

Applicationcontext 继承BeanFactory的接口 更高级的容器 提供更多功能

AOP 消息发送

最大区别

Applicationcontext 启动就把所有bean全部实例化 beanfactory是需要再去实例化bean

有点类似懒汉模式和饿汉模式

一般都用Applicationcontext

SpringMVC可以帮助我们进行更加简洁的Web层开发

流程说明

客户端发送请求，直接请求到DispatcherServlet 会解析请求对应的controller控制器

然后调用处理器完成业务逻辑 处理完业务后 会返回一个modelView对象

然后进行视图渲染 返回给客户端

**Spring框架中用到了哪些设计模式**

1.工厂设计模式：Spring使用工厂模式通过BeanFactory和ApplicationContext创建bean对象。

2.代理设计模式：Spring AOP功能的实现。

3.单例设计模式：Spring中的bean默认都是单例的。

4.模板方法模式：Spring中的jdbcTemplate、hibernateTemplate等以Template结尾的对数据库操作的类，它们就使用到了模板模式。

5.包装器设计模式：我们的项目需要连接多个数据库，而且不同的客户在每次访问中根据需要会去访问不同的数据库。这种模式让我们可以根据客户的需求能够动态切换不同的数据源。

6.观察者模式：Spring事件驱动模型就是观察者模式很经典的一个应用。

7.适配器模式：Spring AOP的增强或通知（Advice）使用到了适配器模式、Spring MVC中也是用到了适配器模式适配Controller。

**@Component和@Bean的区别是什么**

1.作用对象不同。@Component注解作用于类，而@Bean注解作用于方法。

2.@Component注解通常是通过类路径扫描来自动侦测以及自动装配到Spring容器中（我们可以使用@ComponentScan注解定义要扫描的路径）。@Bean注解通常是在标有该注解的方法中定义产生这个bean，告诉Spring这是某个类的实例，当我需要用它的时候还给我。

3.@Bean注解比@Component注解的自定义性更强，而且很多地方只能通过@Bean注解来注册bean。比如当引用第三方库的类需要装配到Spring容器的时候，就只能通过@Bean注解来实现。

**将一个类声明为Spring的bean的注解有哪些？**

我们一般使用@Autowired注解去自动装配bean。而想要把一个类标识为可以用@Autowired注解自动装配的bean，可以采用以下的注解实现：

1.@Component注解。通用的注解，可标注任意类为Spring组件。如果一个Bean不知道属于哪一个层，可以使用@Component注解标注。

2.@Repository注解。对应持久层，即Dao层，主要用于数据库相关操作。

3.@Service注解。对应服务层，即Service层，主要涉及一些复杂的逻辑，需要用到Dao层（注入）。

4.@Controller注解。对应Spring MVC的控制层，即Controller层，主要用于接受用户请求并调用Service层的方法返回数据给前端页面。

**Spring事务管理的方式有几种？**

1.编程式事务：在代码中硬编码（不推荐使用）。

2.声明式事务：在配置文件中配置（推荐使用），分为基于XML的声明式事务和基于注解的声明式事务。

前面copy部分了解即可

现在开始两个必问知识点

五大隔离级别和七大传播事务

ISOLATION\_

DEFAULT：默认 mysql默认采用repeatable\_read （具体去mysql上面可以学到）oracle 采用read\_committed隔离级别

Read\_uncommited 最低隔离 可能会导致三种错误的发送

Read comitted 阻止脏读 其他两种都可能发送

Repectbale\_read 只可以阻止脏读和不可重复读 幻读有可能会发送

Seriallizable 最高隔离级别 遵从ACID 隔离级别 可防止脏读 不可重复读 以及幻读

事务的传播途径

七种

Required 默认 最经常用到的 假如一个类引用其他类的方法 加了事务 如果当前类有事务 引用的类加入该事务 没有事务 就创建一个新的事务

Requires new 比如 买菜 10元 只能买萝卜 买不了土豆 那就创建一个新的事务 把当前事务挂起

（最常用的用的两类）

Supports

如果当前有事务 就加入该事务 没有的话就普通方式运行

Nosupport

以非事务运行 如果有事务 就挂起

Never 非事务运行 如果存在事务的话异常

Mantatory 有事务 运行 没有的话抛异常

Nested 当前存在事务创建一个事务作为嵌套 没有事务就创建一个新的事务