|  |
| --- |
| 技术问题(至少50个) |
| **问题1：对Spring事务的理解**  回答：事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组  如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点.为了确保要么执行,要么不执行,就可以使用事务. 要将有组语句作为事务考虑，就需要通过ACID测试:即原子性,一致性,隔离性和持久性。  **问题2：Spring概述**  回答：Spring是完全面向接口的设计,降低程序耦合性,主要是事务控制并创建bean实例对象.  在SSM整合时,充当黏合剂的作用(把它当做一个大管家，来帮我们管理对象，控制事务等等).Spring框架有两个核心的特性:  1.IOC(Inversion of Control)控制反转/依赖注入,又称DI(Dependency Injection)(依赖注入)  2.AOP 面向切面编程是对OOP(面向对象编程)的延续和补充,我们在项目中主要使用AOP应用于日志记录,性能统计,安全控制,事务处理等方面.  **问题3: IOC & DI**  回答：IOC不是一种技术，只是一种思想，一个重要的面向对象编程的法则，它能指导我们如何设计出松耦合、更优良的程序。传统应用程序都是由我们在类内部主动创建依赖对象，从而导致类与类之间高耦合，难于测试；有了IOC容器后，把创建和查找依赖对象的控制权交给了容器，由容器进行注入组合对象，所以对象与对象之间是 松散耦合，这样也方便测试，利于功能复用，更重要的是使得程序的整个体系结构变得非常灵活。  DI—Dependency Injection，即“依赖注入”：组件之间依赖关系由容器在运行期决定，形象的说，即由容器动态的将某个依赖关系注入到组件之中。依赖注入的目的并非为软件系统带来更多功能，而是为了提升组件重用的频率，并为系统搭建一个灵活、可扩展的平台。通过依赖注入机制，我们只需要通过简单的配置，而无需任何代码就可指定目标需要的资源，完成自身的业务逻辑，而不需要关心具体的资源来自何处，由谁实现。  IOC和DI由什么关系呢？其实它们是同一个概念的不同角度描述，“依赖注入”明确描述了“被注入对象依赖IOC容器配置依赖对象”。  **问题4:** **AOP的理解**  回答：面向切面的编程AOP，是一种编程技术，允许程序模块化横向切割关注点，或横切典型的责任划分，如日志和事务管理。  AOP的核心是切面，它将多个类的通用行为封装成可重用的模块，该模块含有一组API提供横切功能。  在SpringAOP中，切面通过带有@Aspect注解的类实现。 关注点是应用中的一个模块的行为，一个关注点可能会被定义成一个我们想实现的一个功能。 横切关注点一个关注点，此关注点是整个应用都会使用的功能，并影响整个应用，比如日志，安全和数据传输，几乎应用的每个模块都需要的功能。因此这些都属于横切关注点。 连接点代表一个应用程序的某个位置，在这个位置我们可以插入一个AOP切面，它实际上是个应用程序执行Spring AOP的位置。 切点是一个或一组连接点，通知将在这些位置执行。可以通过表达式或匹配的方式指明切入点。 引入运行我们在已存在的类中添加新的方法和属性。  **问题5:** **Spring的优点**  回答：降低了组件之间的耦合性 ，实现了软件各层之间的解耦  可以使用容易提供的众多服务，如事务管理，消息服务等  容器提供单例模式支持  容器提供了AOP技术，利用它很容易实现如权限拦截，运行期监控等功能  容器提供了众多的辅助类，能加快应用的开发  Spring对于主流的应用框架提供了集成支持，如hibernate，JPA，Struts等  Spring属于低侵入式设计，代码的污染极低  独立于各种应用服务器  Spring的DI机制降低了业务对象替换的复杂性  Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可以自由选择Spring的部分或全部.  **问题6:** **请解释Spring Bean的生命周期**  回答：在一个bean实例被初始化时，需要执行一系列的初始化操作以达到可用的状态。同样的，当一个bean不在被调用时需要进行相关的析构操作，并从bean容器中移除。Spring bean factory 负责管理在spring容器中被创建的bean的生命周期。  Bean的生命周期由两组回调（callback）方法组成。  初始化之后调用的回调方法。  销毁之前调用的回调方法。  **问题7:** **Spring框架的优点**  回答：  •轻量：Spring是轻量级的，基本的版本大小为2MB。  •控制反转：Spring通过控制反转实现了松散耦合，对象们给出它们的依赖，而不是创建或查找依赖的对象们。  •面向切面的编程AOP:Spring支持面向切面的编程，并且把应用业务逻辑和系统服务分开。  •容器：Spring包含并管理应用中对象的生命周期和配置  •MVC框架： Spring-MVC。  •事务管理：Spring提供一个持续的事务管理接口，可以扩展到上至本地事务下至全局事务JTA。  •异常处理：Spring提供方便的API把具体技术相关的异常。  **问题8: Spring中自动装配的方式有哪些？**  回答:  - no：不进行自动装配，手动设置Bean的依赖关系.  - byName：根据Bean的名字进行自动装配。  - byType：根据Bean的类型进行自动装配。  - constructor：类似于byType，不过是应用于构造器的参数，如果正好有一个Bean与构造器的参数类型相同则可以自动装配，否则会导致错误。  - autodetect：如果有默认的构造器，则通过constructor的方式进行自动装配，否则使用byType的方式进行自动装配。  **问题9: Spring框架中的单例Beans是线程安全的么**  回答: Spring框架并没有对单例bean进行任何多线程的封装处理。关于单例bean的线程安全和并发问题需要开发者自行去搞定。但实际上，大部分的Spring bean并没有可变的状态(比如Serview类和DAO类)，所以在某种程度上说Spring的单例bean是线程安全的。如果你的bean有多种状态的话（比如ViewModel对象），就需要自行保证线程安全。最浅显的解决办法就是将多态bean的作用域由"singleton"变更为"prototype".  **问题9:** **@Value**  回答: @Value是一个新的装配注解，可以让我们使用注解装配String类型的值和基本类型的值，如int, boolean。我们可以通过@Value直接标注某个属性，方法或者方法参数，并传入一个String类型的表达式来装配属性.  **问题10:Eureka**  回答: Eureka是Netflix开发的服务发现框架，主要用于定位运行在AWS(亚马逊公司旗下云计算服务平台)域中的中间层服务，以达到负载均衡和中间层服务故障转移的目的。  Eureka包含两个组件：Eureka Server和Eureka Client。  Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server中进行注册，这样EurekaServer中的服务注册表中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到。  Eureka Client是一个java客户端，用于简化与Eureka Server的交互，客户端同时也就是一个内置的、使用轮询(round-robin)负载算法的负载均衡器。  ....... |
| **人资问题(至少30个)** |
| 问题1：你得离职原因是啥  回答：单位有工作人员是领导的亲戚或朋友,而自己则是应聘进去的,很多好的机会总轮不到自己  回答2：为什么的年龄和身份证不一致呢？  回答：  ........ |