**Spring 面试**

**Spring 概述**

什么是 spring?

Spring是一个轻量级Java开发框架，最早由Rod Johnson创建，目的是为了解决企业级应用开发的业务逻辑层和其他各层的耦合问题。它是一个分层的JavaSE/JavaEE full-stack（一站式）轻量级开源框架，为开发Java应用程序提供全面的基础架构支持。Spring负责基础架构，因此Java开发者可以专注于应用程序的开发。Spring最根本的使命是解决企业级应用开发的复杂性，即简化Java开发。Spring可以做很多事情，它为企业级开发提供给了丰富的功能，但是这些功能的底层都依赖于它的两个核心特性，也就是依赖注入（dependency injection，DI）和面向切面编程（aspectoriented programming，AOP）。

为了降低Java开发的复杂性，Spring采取了以下4种关键策略：

* 基于POJO的轻量级和最小侵入性编程；
* 通过依赖注入和面向接口实现松耦合；
* 基于切面和惯例进行声明式编程；
* 通过切面和模板减少样板式代码。

Spring是一个分层的JavaSE/EE full-stack(一站式)轻量级开源框架.  
Spring为不同的层都提供的企业级解决方案:  
    web层:spring mvc  
    service层:spring  
    dao层:JDBCTemplate、Spring data jpa(Java Persistence API)  
Spring的核心思想是IOC(控制反转)和AOP(面向切面编程)

Spring 框架的设计目标，设计理念，和核心是什么

Spring设计目标：Spring为开发者提供一个一站式轻量级应用开发平台；

Spring设计理念：在JavaEE开发中，支持POJO和JavaBean开发方式，使应

用面向接口开发，充分支持OO（面向对象）设计方法；Spring通过IoC容器

实现对象耦合关系的管理，并实现依赖反转，将对象之间的依赖关系交给IoC

容器，实现解耦；

Spring框架的核心：IoC容器和AOP模块。通过IoC容器管理POJO对象以及

他们之间的耦合关系；通过AOP以动态非侵入的方式增强服务。

IoC让相互协作的组件保持松散的耦合，而AOP编程允许你把遍布于应用各层

的功能分离出来形成可重用的功能组件。

Spring 的优缺点是什么？

优点：方便解耦，简化开发

（低耦合，从字面上来看有藕断丝连的意思，也就是尽可能的使每个模块之间或者每个类之间再或者是每个方法之间的关联关系减少，这样可以使各自尽可能的独立，一个地方出错，不会影响全部，更能提高代码的重用性）

Spring就是一个大工厂，可以将所有对象的创建和依赖关系的维护，交给Spring管理。

AOP编程的支持，Spring提供面向切面编程，可以方便的实现对程序进行权限拦截、运行监控等功能。

声明式事务的支持，只需要通过配置就可以完成对事务的管理，而无需手动编程。

方便程序的测试，Spring对Junit4支持，可以通过注解方便的测试Spring程序。

方便集成各种优秀框架，Spring不排斥各种优秀的开源框架，其内部提供了对各种优秀框架的直接支持（如：Struts、Hibernate、MyBatis等）。

降低JavaEE API的使用难度，Spring对JavaEE开发中非常难用的一些API（JDBC、JavaMail、远程调用等），都提供了封装，使这些API应用难度大大降低。

1.降低了组件之间的耦合性,实现了软件各层之间的解耦.  
2.可以使用容器提供的众多服务,如事务管理,消息服务等.  
3.容器提供单利模式支持.  
4.容器提供了AOP技术,利用它可以很容易实现的一些拦截,如权限拦截,运行期监控等.  
5.容器提供了AOP技术,利用它可以很容易实现运行拦截,如权限拦截,运行期监控等.  
6.spring对于主流的应用框架提供了很好的支持,例如mybatis等.  
7.spring属于低入侵设计  
8.独立各种应用服务器  
9.spring的DI机制减低了业务对象替换的复杂性.  
10.spring的高开放性,并不强制应用于完全依赖于它,开发者可以自由选择spring的部分或者全部.

**缺点：**

Spring明明一个很轻量级的框架，却给人感觉大而全

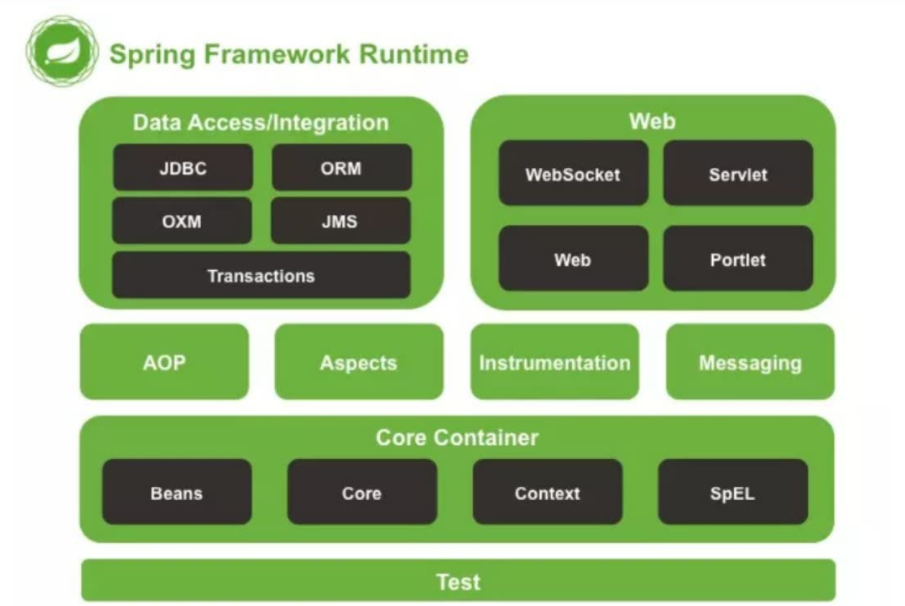
Spring依赖反射，反射影响性能

使用门槛升高，入门Spring需要较长时间

使用人数不多、jsp中要写很多代码、控制器过于灵活，缺少一个公用控制器

Spring 由哪些模块组成？

Spring总共大约有 20个模块，由 1300多个不同的文件构成。 而这些组件被分别整合在核心容器（Core Container）、AOP（Aspect Oriented Programming）和设备支持（Instrmentation）、数据访问与集成（Data Access/Integeration）、Web、消息（Messaging）、Test等6个模块中。以下是 Spring5的模块结构图：



spring core：提供了框架的基本组成部分，包括控制反转（Inversion of Control，IOC）和依赖注入（Dependency Injection，DI）功能。

springbeans：提供了BeanFactory，是工厂模式的一个经典实现，Spring将管理对象称为Bean。

spring context：构建于 core封装包基础上的 context封装包，提供了一种框架式的对象访问方法。

spring jdbc：提供了一个JDBC的抽象层，消除了烦琐的JDBC编码和数据库厂商特有的错误代码解析，用于简化JDBC。

spring aop：提供了面向切面的编程实现，让你可以自定义拦截器、切点等。springWeb：提供了针对Web开发的集成特性，例如文件上传，利用 servlet listeners 进行 ioc 容器初始化和针对Web的 ApplicationContext。

spring test：主要为测试提供支持的，支持使用JUnit或TestNG对Spring组件进行单元测

**1、Spring core：核心容器**

核心容器提供spring框架的基本功能。Spring以bean的方式组织和管理Java应用中的各个组件及其关系。Spring使用BeanFactory来产生和管理Bean，它是工厂模式的实现。BeanFactory使用控制反转(IoC)模式将应用的配置和依赖性规范与实际的应用程序代码分开。BeanFactory使用依赖注入的方式提供给组件依赖。主要实现控制反转IoC和依赖注入DI、Bean配置以及加载。

**2、Spring AOP：Spring面向切面编程**

通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了Spring框架中。所以，可以很容易地使 Spring框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。AOP把一个业务流程分成几部分，例如权限检查、业务处理、日志记录，每个部分单独处理，然后把它们组装成完整的业务流程。每个部分被称为切面或关注点。

AOP的实现原理为动态代理技术，一共有两种代理模式：

（1）ProxyFactoryBean代理工厂对象

Spring内置代理类，引入一个中间层，能够创建不同类型的对象，利用它可以实现任何形式的AOP。

（2）TransactionProxyFactoryBean事务代理工厂对象

常用在数据库编程上，Spring利用TransactionProxyFactoryBean对事务进行管理，在指定方法前利用AOP连接数据库并开启事务，然后在指定方法返回后利用AOP提交事务并断开数据库。

**3、Spring context：Spring上下文**

Spring上下文是一个配置文件，向Spring框架提供上下文信息。Spring上下文包括企业服务，如JNDI、EJB、电子邮件、国际化、校验和调度功能。提供框架式Bean访问方式，其他程序可以通过Context访问Spring的Bean资源。

**4、Spring DAO**

DAO模块主要目的是将持久层相关问题与一般的的业务规则和工作流隔离开来。Spring 中的DAO提供一致的方式访问数据库，不管采用何种持久化技术，Spring都提供一致的编程模型。Spring还对不同的持久层技术提供一致的DAO方式的异常层次结构。Spring的DAO模块对JDBC进行了再封装，隐藏了Connection、Statement、ResultSet等JDBC API，使DAO模块直接继承JdbcDaoSupport类。

**5、Spring ORM（Object Relation Mapper）对象关系映射模块**

Spring 与所有的主要的ORM框架都集成的很好，包括hibernate、JDO实现、TopLink和IBatis SQL Map等。Spring为所有的这些框架提供了模板之类的辅助类，达成了一致的编程风格。  
Spring的ORM模块对ORM框架如Hibernate等进行了封装，Spring能够管理、维护Hibernate，使用时可直接继承HibernateDaoSupport类，该类内置一个HibernateTemplate。Hibernate的配置也转移到Spring配置文件中。

（注：ORM是通过使用描述对象和数据库之间映射的元数据，ORM框架采用元数据来描述对象--关系映射细节，元数据一般采用xml格式，并且存放在专门的对象--映射文件中）

**6、Spring Web模块**

Web模块建立在应用程序上下文模块之上，为基于Web的应用程序提供了上下文。Web层使用Web层框架，可选的，可以是Spring自己的MVC框架，或者提供的Web框架，如Struts、Webwork、tapestry和jsf。

Web模块用于整合Web框架，将Web框架也纳入Spring的管理之中。如Spring提供继承方式与代理方式整合Struts，继承方式不需要更改任何配置文件，只把Action继承自ActionSupport即可，但会对Spring产生依赖。代理方式需要在struts-config.xml中配置<controller>，由Spring全盘代理，因此可以使用Spring的各种资源、拦截器等。

**7、Spring MVC**

MVC框架是一个全功能的构建Web应用程序的MVC实现。通过策略接口，MVC框架变成为高度可配置的。Spring的MVC框架提供清晰的角色划分：控制器、验证器、命令对象、表单对象和模型对象、分发器、处理器映射和视图解析器。Spring支持多种视图技术。

Spring MVC 的工作流程：

（1） 客户端发送请求，请求到达 DispatcherServlet 主控制器。  
（2） DispatcherServlet 控制器调用 HandlerMapping 处理。  
（3） HandlerMapping 负责维护请求和 Controller 组件对应关系。 HandlerMapping 根据请求调用对应的 Controller 组件处理。  
（4） 执行 Controller 组件的业务处理，需要访问数据库，可以调用 DAO 等组件。  
（5）Controller 业务方法处理完毕后，会返回一个 ModelAndView 对象。该组件封装了模型数据和视图标识。  
（6）Servlet 主控制器调用 ViewResolver 组件，根据 ModelAndView 信息处理。定位视图资源，生成视图响应信息。  
（7）控制器将响应信息给用户输出。

Spring 框架中都用到了哪些设计模式？

**工厂设计模式** : BeanFactory就是简单工厂模式的体现，用来创建对象的实例；Spring使用工厂模式通过 BeanFactory、ApplicationContext 创建 bean 对象。

**代理设计模式** : Spring AOP 功能的实现。Spring的AOP功能用到了JDK的动态代理和CGLIB字节码生成技术；

**单例设计模式** : Spring 中的 Bean 默认都是单例的。

**模板方法模式** : Spring 中 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。用来解决代码重复的问题。比如.RestTemplate,JmsTemplate,JpaTemplate。

**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。

**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。定义对象键一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都会得到通知被制动更新，如Spring中listener的实现–ApplicationListener。

**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。

详细讲解一下核心容器（spring context 应用上下文) 模块

这是基本的Spring模块，提供spring 框架的基础功能，BeanFactory 是任何以spring为基础的应用的核心。Spring 框架建立在此模块之上，它使Spring成为一个容器。Bean 工厂是工厂模式的一个实现，提供了控制反转功能，用来把应用的配置和依赖从真正的应用代码中分离。最常用的就是org.springframework.beans.factory.xml.XmlBeanFactory ，它根据XML文件中的定义加载beans。该容器从XML文件读取配置元数据并用它去创建一个完全配置的系统或应用。

Spring 应用程序有哪些不同组件？

Spring 应用一般有以下组件：

接口 -定义功能。

Bean类- 它包含属性，setter 和 getter 方法，函数等。

Bean配置文件 -包含类的信息以及如何配置它们。

Spring 面向切面编程（AOP）- 提供面向切面编程的功能。

用户程序 - 它使用接口。

**Spring 控制反转(IOC)**

什么是 Spring IOC 容器？

控制反转即IoC (Inversion of Control)，它把传统上由程序代码直接操控的对象的调用权交给容器，通过容器来实现对象组件的装配和管理。所谓的“控制反转”概念就是对组件对象控制权的转移，从程序代码本身转移到了外部容器。Spring IOC 负责创建对象，管理对象（通过依赖注入（DI），装配对象，配置对象，并且管理这些对象的整个生命周期。

控制反转(IoC)有什么作用

管理对象的创建和依赖关系的维护。对象的创建并不是一件简单的事，在对象关系比较复杂时，如果依赖关系需要程序猿来维护的话，那是相当头疼的解耦，由容器去维护具体的对象托管了类的产生过程，比如我们需要在类的产生过程中做一些处理，最直接的例子就是代理，如果有容器程序可以把这部分处理交给容器，应用程序则无需去关心类是如何完成代理的

Spring IoC 的实现机制





BeanFactory 和 ApplicationContext 有什么区别？

BeanFactory和ApplicationContext是Spring的两大核心接口，都可以当做Spring的容器。其中ApplicationContext是BeanFactory的子接口。

**依赖关系BeanFactory：**是Spring里面最底层的接口，包含了各种Bean的定义，读取bean配置文档，管理bean的加载、实例化，控制bean的生命周期，维护bean之间的依赖关系。ApplicationContext接口作为BeanFactory的派生，除了提供BeanFactory所具有的功能外，还提供了更完整的框架功能：

继承MessageSource，因此支持国际化。

统一的资源文件访问方式。

提供在监听器中注册bean的事件。

同时加载多个配置文件。

载入多个（有继承关系）上下文，使得每一个上下文都专注于一个特定的层次，比如应用的web层。

**加载方式：**

BeanFactroy采用的是延迟加载形式来注入Bean的，即只有在使用到某个Bean时(调用getBean())，才对该Bean进行加载实例化。这样，我们就不能发现一些存在的Spring的配置问题。如果Bean的某一个属性没有注入，BeanFacotry加载后，直至第一次使用调用getBean方法才会抛出异常。ApplicationContext，它是在容器启动时，一次性创建了所有的Bean。这样，在容器启动时，我们就可以发现Spring中存在的配置错误，这样有利于检查所依赖属性是否注入。ApplicationContext启动后预载入所有的单实例Bean，通过预载入单实例bean,确保当你需要的时候，你就不用等待，因为它们已经创建好了。相对于基本的BeanFactory，ApplicationContext唯一的不足是占用内存空间。当应用程序配置Bean较多时，程序启动较慢。

**创建方式：**BeanFactory通常以编程的方式被创建，ApplicationContext还能以声明的方式创建，如使用ContextLoader。

**注册方式：**

BeanFactory和ApplicationContext都支持BeanPostProcessor、BeanFactoryPostProcessor的使用，但两者之间的区别是：BeanFactory需要手动注册，而ApplicationContext则是自动注册

什么是 Spring 的依赖注入？

控制反转IoC是一个很大的概念，可以用不同的方式来实现。其主要实现方式有两种：依赖注入和依赖查找依赖注入：相对于IoC而言，依赖注入(DI)更加准确地描述了IoC的设计理念。所谓依赖注入（Dependency Injection），即组件之间的依赖关系由容器在应用系统运行期来决定，也就是由容器动态地将某种依赖关系的目标对象实例注入到应用系统中的各个关联的组件之中。组件不做定位查询，只提供普通的Java方法让容器去决定依赖关系。

依赖注入的基本原则

依赖注入的基本原则是：应用组件不应该负责查找资源或者其他依赖的协作对象。配置对象的工作应该由IoC容器负责，“查找资源”的逻辑应该从应用组件的代码中抽取出来，交给IoC容器负责。容器全权负责组件的装配，它会把符合依赖关系的对象通过属性（JavaBean中的setter）或者是构造器传递给需要的对象。

有哪些不同类型的依赖注入实现方式？

依赖注入是时下最流行的IoC实现方式，依赖注入分为接口注入（InterfaceInjection），Setter方法注入（Setter Injection）和构造器注入（Constructor Injection）三种方式。其中接口注入由于在灵活性和易用性比较差，现在从Spring4开始已被废弃。构造器依赖注入：构造器依赖注入通过容器触发一个类的构造器来实现的，该类有一系列参数，每个参数代表一个对其他类的依赖。Setter方法注入：Setter方法注入是容器通过调用无参构造器或无参static工厂 方法实例化bean之后，调用该bean的setter方法，即实现了基于setter的依赖注入。

Spring Beans

什么是 Spring beans？

Springbeans 是那些形成Spring应用的主干的java对象。它们被Spring IOC容器初始化，装配，和管理。这些beans通过容器中配置的元数据创建。比如，以XML文件中 的形式定义。

如何给 Spring 容器提供配置元数据？Spring 有几种配置方式

这里有三种重要的方法给Spring 容器提供配置元数据。

XML配置文件。基于注解的配置。基于java的配置。

Spring 配置文件包含了哪些信息

Spring配置文件是个XML文件，这个文件包含类信息，描述了如何配置它们，以及如何相互调用。

Spring 基于 xml 注入 bean 的几种方式

Set方法注入；

构造器注入：①通过index设置参数的位置；②通过type设置参数类型；

静态工厂注入；

实例工厂；

你怎样定义类的作用域？

当定义一个bean在Spring里，我们还能给这个bean声明一个作用域。它可以通过bean定义中的scope属性来定义。如，当Spring要在需要的时候每次生产一个新的bean实例，bean的scope属性被指定为prototype。另一方面，一个bean每次使用的时候必须返回同一个实例，这个bean的scope属性必须设为 singleton。

解释 Spring 支持的几种 bean 的作用域

Spring框架支持以下五种bean的作用域：singleton:bean在每个Spring ioc 容器中只有一个实例。prototype：一个bean的定义可以有多个实例。request：每次http请求都会创建一个bean，该作用域仅在基于web的SpringApplicationContext情形下有效。session：在一个HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext情形下有效。global-session：在一个全局的HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext情形下有效。注意：缺省的Springbean 的作用域是Singleton。使用 prototype 作用域需要慎重的思考，因为频繁创建和销毁bean会带来很大的性能开销

Spring 框架中的单例 bean 是线程安全的吗？

不是，Spring框架中的单例bean不是线程安全的。spring 中的bean默认是单例模式，spring 框架并没有对单例bean 进行多线程的封装处理。实际上大部分时候springbean无状态的（比如dao类），所有某种程度上来说bean 也是安全的，但如果bean 有状态的话（比如viewmodel 对象），那就要开发者自己去保证线程安全了，最简单的就是改变bean 的作用域，把“singleton”变更为“prototype”，这样请求bean 相当于 newBean()了，所以就可以保证线程安全了。有状态就是有数据存储功能。无状态就是不会保存数据。

什么是 Spring 的内部 bean？什么是 Spring inner beans？

在Spring框架中，当一个bean仅被用作另一个bean的属性时，它能被声明为一个内部bean。内部bean可以用setter注入“属性”和构造方法注入“构造参数”的方式来实现，内部bean通常是匿名的，它们的Scope一般是prototype。

什么是 bean 装配？

装配，或bean装配是指在Spring 容器中把bean组装到一起，前提是容器需要知道bean的依赖关系，如何通过依赖注入来把它们装配到一起。

什么是 bean 的自动装配？

在Spring框架中，在配置文件中设定bean的依赖关系是一个很好的机制，Spring 容器能够自动装配相互合作的bean，这意味着容器不需要和配置，能通过Bean工厂自动处理bean之间的协作。这意味着Spring可以通过向BeanFactory中注入的方式自动搞定bean之间的依赖关系。自动装配可以设置在每个bean上，也可以设定在特定的bean上。

解释不同方式的自动装配，spring 自动装配 bean 有哪些方式？

在spring中，对象无需自己查找或创建与其关联的其他对象，由容器负责把需要相互协作的对象引用赋予各个对象，使用autowire来配置自动装载模式。在Spring框架xml配置中共有5种自动装配：no：默认的方式是不进行自动装配的，通过手工设置ref属性来进行装配bean。byName：通过bean的名称进行自动装配，如果一个bean的 property与另一bean 的name相同，就进行自动装配。

byType：通过参数的数据类型进行自动装配。

constructor：利用构造函数进行装配，并且构造函数的参数通过byType进行装配。

autodetect：自动探测，如果有构造方法，通过 construct的方式自动装配，否则使用byType的方式自动装配。

使用@Autowired 注解自动装配的过程是怎样的？

使用@Autowired注解来自动装配指定的bean。在使用@Autowired注解之前需要在Spring配置文件进行配置，<context:annotation-config />。在启动spring IoC时，容器自动装载了一个AutowiredAnnotationBeanPostProcessor后置处理器，当容器扫描到@Autowied、@Resource或@Inject时，就会在IoC容器自动查找需要的bean，并装配给该对象的属性。在使用@Autowired时，首先在容器中查询对应类型的bean：如果查询结果刚好为一个，就将该bean装配给@Autowired指定的数据；如果查询的结果不止一个，那么@Autowired会根据名称来查找；如果上述查找的结果为空，那么会抛出异常。解决方法时，使用required=false。

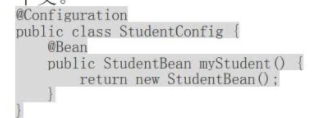
你可以在 Spring 中注入一个 null 和一个空字符串吗？

可以

**Spring 注解**

什么是基于 Java 的 Spring 注解配置? 给一些注解的例子

基于Java的配置，允许你在少量的Java注解的帮助下，进行你的大部分Spring配置而非通过XML文件。以@Configuration 注解为例，它用来标记类可以当做一个bean的定义，被Spring IOC容器使用。另一个例子是@Bean注解，它表示此方法将要返回一个对象，作为一个bean注册进Spring应用上下文。



怎样开启注解装配？

注解装配在默认情况下是不开启的，为了使用注解装配，我们必须在Spring配置文件中配置<context:annotation-config/>元素。

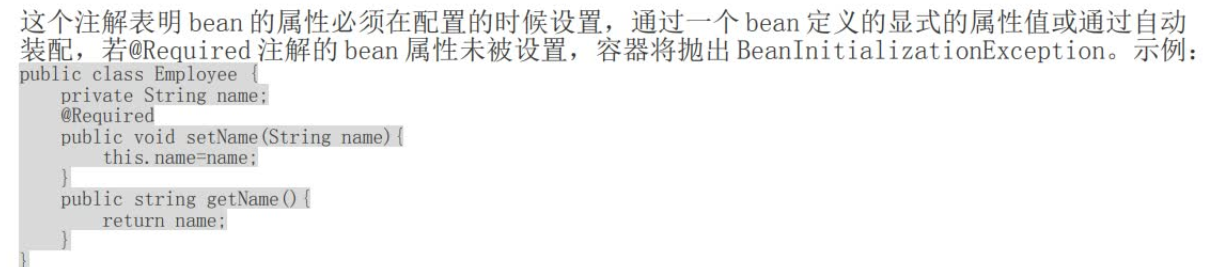
@Component, @Controller, @Repository, @Service 有 何 区

别？

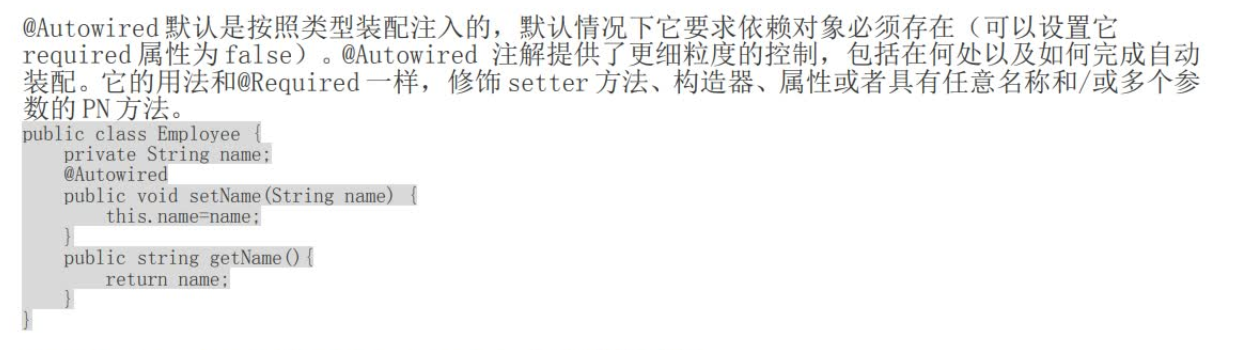
@Component：这将 java类标记为bean。它是任何Spring 管理组件的通用构造型。spring 的组件扫描机制现在可以将其拾取并将其拉入应用程序环境中。@Controller：这将一个类标记为 SpringWebMVC 控制器。标有它的 Bean会自动导入到IoC容器中。

@Service：此注解是组件注解的特化。它不会对@Component 注解提供任何其他行为。您可以在服务层类中使用@Service 而不是@Component，因为它以更好的方式指定了意图。@Repository：这个注解是具有类似用途和功能的@Component 注解的特化。它为 DAO 提供了额外的好处。它将 DAO导入 IoC 容器，并使未经检查的异常有资格转换为 SpringDataAccessException。

@Required 注解有什么作用



@Autowired 注解有什么作用



@Autowired 和@Resource 之间的区别

@Autowired可用于：构造函数、成员变量、Setter方法@Autowired和@Resource之间的区别@Autowired默认是按照类型装配注入的，默认情况下它要求依赖对象必须存在（可以设置它required属性为false）。@Resource默认是按照名称来装配注入的，只有当找不到与名称匹配的bean才会按照类型来装配注入。

@Qualifier 注解有什么作用

当您创建多个相同类型的bean 并希望仅使用属性装配其中一个bean时，您可以使用@Qualifier注解和@Autowired 通过指定应该装配哪个确切的bean 来消除歧义。

@RequestMapping 注解有什么用？

@RequestMapping 注解用于将特定HTTP请求方法映射到将处理相应请求的控制器中的特定类/方法。此注释可应用于两个级别：类级别：映射请求的URL方法级别：映射URL以及HTTP请求方法

**Spring 数据访问**

Spring 支持的事务管理类型， spring 事务实现方式有哪些？

Spring支持两种类型的事务管理：

* 编程式事务管理：这意味你通过编程的方式管理事务，给你带来极大的灵活性，但是难维护。
* 声明式事务管理：这意味着你可以将业务代码和事务管理分离，你只需用注解和XML配置来管理事务。

Spring 事务的实现方式和实现原理

Spring事务的本质其实就是数据库对事务的支持，没有数据库的事务支持，spring是无法提供事务功能的。真正的数据库层的事务提交和回滚是通过binlog或者redo log实现的。

说一下 spring 的事务隔离？

spring 有五大隔离级别，默认值为 ISOLATIONDEFAULT（使用数据库的设置），其他四个隔离级别和数据库的隔离级别一致：1ISOLATIONDEFAULT：(default--默认)用底层数据库的设置隔离级别，数据库设置的是什么我就用什么；2ISOLATIONREADUNCOMMITTED：(read uncommitted--未遂的)未提交读，最低隔离级别、事务未提交前，就可被其他事务读取（会出现幻读、脏读、不可重复读）；3ISOLATION\_READ\_COMMITTED：(read committed--付诸的)提交读，一个事务提交后才能被其他事务读取到（会造成幻读、不可重复读），SQLserver 的默认级别；4ISOLATION\_REPEATABLE\_READ：(repeatable read--可重复)可重复读，保证多次读取同一个数据时，其值都和事务开始时候的内容是一致，禁止读取到别的事务未提交的数据（会造成幻读），MySQL的默认级别；5ISOLATIONSERIALIZABLE：(serializable--序列化)序列化，代价最高最可靠的隔离级别，该隔离级别能防止脏读、不可重复读、幻读。幻读：指同一个事务内多次查询返回的结果集不一样。比如同一个事务 A第一次查询时候有n条记录，但是第二次同等条件下查询却有 n+1条记录，这就好像产生了幻觉。发生幻读的原因也是另外一个事务新增或者删除或者修改了第一个事务结果集里面的数据，同一个记录的数据内容被修改了，所有数据行的记录就变多或者变少了。侧重与新增或者删除。办法锁表。不可重复读：是指在一个事务内，多次读同一数据,结果不一致。侧重于修改。办法锁行脏读：表示一个事务能够读取另一个事务中还未提交的数据。比如，某个事务尝试插入记录A，此时该事务还未提交，然后另一个事务尝试读取到了记录A。

你更倾向用那种事务管理类型？

**Spring 面向切面编程(AOP)**

什么是 AOP

OOP(Object-Oriented Programming)面向对象编程，允许开发者定义纵向的关系，但并适用于定义横向的关系，导致了大量代码的重复，而不利于各个模块的重用。AOP(Aspect-Oriented Programming)，一般称为面向切面编程，作为面向对象的一种补充，用于将那些与业务无关，但却对多个对象产生影响的公共行为和逻辑，抽取并封装为一个可重用的模块，这个模块被命名为“切面”（Aspect），减少系统中的重复代码，降低了模块间的耦合度，同时提高了系统的可维护性。可用于权限认证、日志、事务处理等。

JDK 动态代理和 CGLIB 动态代理的区别

Spring AOP中的动态代理主要有两种方式，JDK动态代理和CGLIB动态代理：JDK动态代理只提供接口的代理，不支持类的代理。核心InvocationHandler接口和Proxy类，InvocationHandler 通过invoke()方法反射来调用目标类中的代码，动态地将横切逻辑和业务编织在一起；接着，Proxy利用 InvocationHandler动态创建一个符合某一接口的的实例,生成目标类的代理对象。如果代理类没有实现 InvocationHandler 接口，那么Spring AOP会选择使用CGLIB来动态代理目标类。CGLIB（CodeGenerationLibrary），是一个代码生成的类库，可以在运行时动态的生成指定类的一个子类对象，并覆盖其中特定方法并添加增强代码，从而实现AOP。CGLIB是通过继承的方式做的动态代理，因此如果某个类被标记为final，那么它是无法使用CGLIB做动态代理的。静态代理与动态代理区别在于生成AOP代理对象的时机不同，相对来说AspectJ的静态代理方式具有更好的性能，但是AspectJ需要特定的编译器进行处理，而Spring AOP则无需特定的编译器处理。InvocationHandler 的 invoke(Object proxy,Method method,Object[]args)：proxy是最终生成的代理实例;method 是被代理目标实例的某个具体方法;args 是被代理目标实例某个方法的具体入参,在方法反射调用时使用。

如何理解 Spring 中的代理？

将 Advice 应用于目标对象后创建的对象称为代理。在客户端对象的情况下，目标对象和代理对象是相同的。Advice+Target Object=Proxy

解释一下 Spring AOP 里面的几个名词

（1）切面（Aspect）：切面是通知和切点的结合。通知和切点共同定义了切面的全部内容。在Spring AOP中，切面可以使用通用类（基于模式的风格）或者在普通类中以@AspectJ 注解来实现。

（2）连接点（Join point）：指方法，在Spring AOP中，一个连接点总是 代表一个方法的执行。应用可能有数以千计的时机应用通知。这些时机被称为连接点。连接点是在应用执行过程中能够插入切面的一个点。这个点可以是调用方法时、抛出异常时、甚至修改一个字段时。切面代码可以利用这些点插入到应用的正常流程之中，并添加新的行为。

（3）通知（Advice）：在AOP术语中，切面的工作被称为通知。

（4）切入点（Pointcut）：切点的定义会匹配通知所要织入的一个或多个连接点。我们通常使用明确的类和方法名称，或是利用正则表达式定义所匹配的类和方法名称来指定这些切点。

（5）引入（Introduction）：引入允许我们向现有类添加新方法或属性。

（6）目标对象（Target Object）：被一个或者多个切面（aspect）所通知（advise）的对象。它通常是一个代理对象。也有人把它叫做被通知（adviced）对象。既然Spring AOP是通过运行时代理实现的，这个对象永远是一个被代理（proxied）对象。

（7）织入（Weaving）：织入是把切面应用到目标对象并创建新的代理对象的过程。在目标对象的生命周期里有多少个点可以进行织入：编译期：切面在目标类编译时被织入。AspectJ的织入编译器是以这种方式织入切面的。类加载期：切面在目标类加载到JVM时被织入。需要特殊的类加载器，它可以在目标类被引入应用之前增强该目标类的字节码。AspectJ5的加载时织入就支持以这种方式织入切面。运行期：切面在应用运行的某个时刻被织入。一般情况下，在织入切面时，AOP容器会为目标对象动态地创建一个代理对象。SpringAOP就是以这种方式织入切面。

Spring 在运行时通知对象

通过在代理类中包裹切面，Spring在运行期把切面织入到Spring管理的bean中。代理封装了目标类，并拦截被通知方法的调用，再把调用转发给真正的目标bean。当代理拦截到方法调用时，在调用目标bean方法之前，会执行切面逻辑。直到应用需要被代理的bean时，Spring才创建代理对象。如果使用的是ApplicationContext的话，在ApplicationContext从BeanFactory中加载所有bean的时候，Spring才会创建被代理的对象。因为Spring运行时才创建代理对象，所以我们不需要特殊的编译器来织入SpringAOP的切面。

Spring 通知有哪些类型？

在AOP术语中，切面的工作被称为通知，实际上是程序执行时要通过SpringAOP框架触发的代码段。Spring切面可以应用5种类型的通知：前置通知（Before）：在目标方法被调用之前调用通知功能；后置通知（After）：在目标方法完成之后调用通知，此时不会关心方法的输出是什么；返回通知（After-returning ）：在目标方法成功执行之后调用通知；异常通知（After-throwing）：在目标方法抛出异常后调用通知；环绕通知（Around）：通知包裹了被通知的方法，在被通知的方法调用之前和调用之后执行自定义的行为。同一个aspect，不同advice的执行顺序：①没有异常情况下的执行顺序：aroundbefore advice，before advice，target method执行，around after advice，afteradvice，afterReturning②有异常情况下的执行顺序：aroundbefore advice，before advice，target method执行，around after advice，afteradvice，afterThrowing:异常发生，java.lang.RuntimeException:异常发生

什么是切面 Aspect？

aspect 由 advice 和pointcount 组成，切面是通知和切点的结合。它既包含了横切逻辑的定义,也包括了连接点的定义.Spring AOP 就是负责实施切面的框架,它将切面所定义的横切逻辑编织到切面所指定的连接点中.AOP 的工作重心在于如何将增强编织目标对象的连接点上,这里包含两个工作:如何通过 pointcut 和 advice定位到特定的 joinpoint上如何在 advice 中编写切面代码.可以简单地认为,使用@Aspect 注解的类就是切面.

