1、什么是Spring框架?Spring框架有哪些主要模块?

Spring是一个轻量级的JavaEE框架，它主要解决企业应用中的复杂性问题。Spring框架有三个核心部分：IoC容器、AOP和数据访问/集成层。Spring中的IoC容器提供了一种对象创建和对象之间关系管理的机制，以实现松散耦合和可扩展性。AOP提供了一种很好的方式来实现横向关注点的处理，如事务管理，安全检查，缓存等。数据访问/集成层则提供了许多针对不同数据持久化技术的实现，比如JDBC，ORM和NoSQL。。

Spring框架本身亦是按照设计模式精心打造，这使得我们可以在开发环境中安心的集成Spring框架，不必担心Spring是如何在后台进行工作的。Spring框架至今已集成了20多个模块。这些模块主要被分如下图所示的核心容器、数据访问/集成，、Web、AOP(面向切面编程)、工具、消息和测试模块

 1710223732632

除了上述内容之外，希望大家对于spring能有一个更加广义的理解，这里大家记住两个名词：生态和基石。

2.spring有哪些优点？

轻量级：Spring在大小和透明性方面绝对属于轻量级的，基础版本的Spring框架大约只有2MB。

控制反转(IOC)：Spring使用控制反转技术实现了松耦合。依赖被注入到对象，而不是创建或寻找依赖对象。

面向切面编程(AOP)： Spring支持面向切面编程，同时把应用的业务逻辑与系统的服务分离开来。

容器：Spring包含并管理应用程序对象的配置及生命周期。

MVC框架：Spring的web框架是一个设计优良的web MVC框架，很好的取代了一些web框架。

事务管理：Spring对下至本地业务上至全局业务(JAT)提供了统一的事务管理接口。

异常处理：Spring提供一个方便的API将特定技术的异常(由JDBC, Hibernate, 或JDO抛出)转化为一致的、Unchecked异常。

扩展性：后续使用的springboot或者springcloud等相关的技术都是在spring基础之上扩展来的，也就是说spring能够经久不衰，最关键的点就在于它的扩展性

3、什么是控制反转(IOC)?什么是依赖注入?

控制反转是应用于软件工程领域中的，在运行时被装配器对象来绑定耦合对象的一种编程技巧，对象之间耦合关系在编译时通常是未知的。在传统的编程方式中，业务逻辑的流程是由应用程序中的早已被设定好关联关系的对象来决定的。在使用控制反转的情况下，业务逻辑的流程是由对象关系图来决定的，该对象关系图由装配器负责实例化，这种实现方式还可以将对象之间的关联关系的定义抽象化。而绑定的过程是通过“依赖注入”实现的。

控制反转是一种以给予应用程序中目标组件更多控制为目的设计范式，并在我们的实际工作中起到了有效的作用。

依赖注入是在编译阶段尚未知所需的功能是来自哪个的类的情况下，将其他对象所依赖的功能对象实例化的模式。这就需要一种机制用来激活相应的组件以提供特定的功能，所以依赖注入是控制反转的基础。否则如果在组件不受框架控制的情况下，框架又怎么知道要创建哪个组件?

在Java中依赖注入有以下三种实现方式：

1.构造器注入

2.Setter方法注入

3.接口注入

4、谈一下你对于springIOC的理解？

Spring的IOC，即Inversion of Control，也就是控制反转，是Spring框架的核心特性之一。其主要思想是将原本在代码中直接操控的对象的调用权交给Spring容器来管理，从而降低了代码之间的耦合度，提高了程序的可维护性和可扩展性。

在传统的Java开发中，我们通常在代码中直接new一个对象来调用其方法。但这种方式会导致代码之间的耦合度过高，一旦某个类发生改变，可能会影响到很多其他的类。而Spring的IOC解决了这个问题，它将对象的创建和生命周期管理交给了Spring容器来负责。我们只需要在配置文件中配置好相应的Bean，然后就可以在需要的地方通过Spring容器来获取这个Bean，而不需要关心它的创建和销毁过程。

整个IOC容器创建的流程和步骤如下：

1、一般聊ioc容器的时候要涉及到容器的创建过程（beanFactory,DefaultListableBeanFactory）,向bean工厂中设置一些参数（BeanPostProcessor,Aware接口的子类）等等属性

2、加载解析bean对象，准备要创建的bean对象的定义对象beanDefinition,(xml或者注解的解析过程)

3、beanFactoryPostProcessor的处理，此处是扩展点，PlaceHolderConfigurSupport,ConfigurationClassPostProcessor

4、BeanPostProcessor的注册功能，方便后续对bean对象完成具体的扩展功能

5、通过反射的方式讲BeanDefinition对象实例化成具体的bean对象，

6、bean对象的初始化过程（填充属性，调用aware子类的方法，调用BeanPostProcessor前置处理方法，调用init-mehtod方法，调用BeanPostProcessor的后置处理方法）

7、生成完整的bean对象，通过getBean方法可以直接获取

8、销毁过程

5、描述下bean的生命周期？

1、实例化bean：反射的方式生成对象

2、填充bean的属性：populateBean(),循环依赖的问题（三级缓存）

3、调用aware接口相关的方法：invokeAwareMethod(完成BeanName,BeanFactory,BeanClassLoader对象的属性设置)

4、调用BeanPostProcessor中的前置处理方法：使用比较多的有（ApplicationContextPostProcessor,设置ApplicationContext,Environment,ResourceLoader,EmbeddValueResolver等对象）

5、调用initmethod方法：invokeInitmethod(),判断是否实现了initializingBean接口，如果有，调用afterPropertiesSet方法，没有就不调用

6、调用BeanPostProcessor的后置处理方法：spring的aop就是在此处实现的，AbstractAutoProxyCreator

注册Destuction相关的回调接口：钩子函数

7、获取到完整的对象，可以通过getBean的方式来进行对象的获取

8、销毁流程，1；判断是否实现了DispoableBean接口，2，调用destroyMethod方法

6、解释下自动装配的各种模式：

自动装配提供五种不同的模式供Spring容器用来自动装配beans之间的依赖注入:

no：默认的方式是不进行自动装配，通过手工设置ref 属性来进行装配bean。

byName：通过参数名自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byName。之后容器试图匹配、装配和该bean的属性具有相同名字的bean。

byType：通过参数的数据类型自动自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byType。之后容器试图匹配和装配和该bean的属性类型一样的bean。如果有多个bean符合条件，则抛出错误。

constructor：这个同byType类似，不过是应用于构造函数的参数。如果在BeanFactory中不是恰好有一个bean与构造函数参数相同类型，则抛出一个严重的错误。

autodetect：如果有默认的构造方法，通过 construct的方式自动装配，否则使用 byType的方式自动装配。

7、BeanFactory和ApplicationContext有什么区别？

1. 加载方式：BeanFactory采用的是延迟加载（lazy-loading）的方式，即只有在使用到某个Bean时才会对该Bean进行加载和实例化。这种方式可以减少程序启动时的内存消耗，但如果在使用Bean之前没有对其进行正确的配置，可能会导致运行时错误。而ApplicationContext则是在容器启动时，一次性创建所有的Bean。这样做的好处是可以在容器启动时就发现Spring配置中的错误，但可能会增加程序启动时的内存消耗。
2. 创建方式：BeanFactory通常以编程的方式创建，需要手动编写代码来配置和获取Bean。而ApplicationContext除了支持编程方式创建外，还支持声明方式创建，如使用XML配置文件或注解来配置Bean。这使得ApplicationContext的配置更加灵活和方便。
3. 功能：ApplicationContext接口作为BeanFactory的派生，不仅提供了BeanFactory所具有的功能，还提供了更完整的框架功能。例如，它继承了MessageSource，支持国际化，统一资源文件访问的方式。此外，ApplicationContext还支持在监听器中注册bean事件，同时加载多个配置文件和载入多个上下文。

8、谈一下你对于spring AOP的理解

面向切面编程（AOP）：允许程序员模块化横向业务逻辑，或定义核心部分的功能，例如日志管理和事务管理。

切面(Aspect) ：AOP的核心就是切面，它将多个类的通用行为封装为可重用的模块。该模块含有一组API提供 cross-cutting功能。例如,日志模块称为日志的AOP切面。根据需求的不同，一个应用程序可以有若干切面。在Spring AOP中，切面通过带有@Aspect注解的类实现。

通知(Advice)：通知表示在方法执行前后需要执行的动作。实际上它是Spring AOP框架在程序执行过程中触发的一些代码。Spring切面可以执行一下五种类型的通知:

- before(前置通知)：在一个方法之前执行的通知。
- after(最终通知)：当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。
- after-returning(后置通知)：在某连接点正常完成后执行的通知。
- after-throwing(异常通知)：在方法抛出异常退出时执行的通知。
- around(环绕通知)：在方法调用前后触发的通知。

切入点(Pointcut)：切入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：引入允许我们在已有的类上添加新的方法或属性。

目标对象：被一个或者多个切面所通知的对象。它通常是一个代理对象。也被称做被通知（advised）对象。

代理：代理是将通知应用到目标对象后创建的对象。从客户端的角度看，代理对象和目标对象是一样的。有以下几种代理：

- BeanNameAutoProxyCreator：bean名称自动代理创建器
- DefaultAdvisorAutoProxyCreator：默认通知者自动代理创建器
- Metadata autoproxying：元数据自动代理

织入：将切面和其他应用类型或对象连接起来创建一个通知对象的过程。织入可以在编译、加载或运行时完成。

9、spring中用到了哪些设计模式

单例模式：bean默认都是单例的

原型模式：指定作用域为prototype

工厂模式：BeanFactory

模板方法：postProcessBeanFactory,onRefresh,initPropertyValue

策略模式：XmlBeanDefinitionReader,PropertiesBeanDefinitionReader

观察者模式：listener，event，multicast

适配器模式：Adapter

装饰者模式：BeanWrapper

责任链模式：使用aop的时候会先生成一个拦截器链

代理模式：动态代理

委托者模式：delegate