1. 你为什么使用 Spring?

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。

2. Spring 支持几种 bean 的作用域?

1)singleton：默认，每个容器中只有一个 bean 的实例，单例的模式由 BeanFactory 自身来维护。

2)prototype：为每一个 bean 请求提供一个实例。

3)request：为每一个网络请求创建一个实例，在请求完成以后，bean 会失效并被垃圾回收器回收。

4)session：与 request 范围类似，确保每个 session 中有一个 bean 的实例，在 session 过期后，bean 会随之失效。

5)global-session：全局作用域，global-session 和 Portlet 应用相关。当你的应用部署在 Portlet 容器中工作时，它包含很多 portlet。如果你想要声明让所有的 portlet 共用全局的存储变量的话，那么这全局变量需要存储在 global-session 中。全局作用域与 Servlet 中的 session 作用域效果相同。

3. 请问 Spring 有几种自动装配模式？

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。


微信搜一搜
享学课堂online

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。

4. 对 Java 接口代理模式的实现原理的理解?

答:

1、字符串拼凑出代理类
2、用流的方式写到.java 文件
3、动态编译.java 文件
4、自定义类加载器把.class 文件加载到 jvm
5、返回代理类实例
其实核心就是根据接口反射出接口中的所有方法,然后通过拼凑或者 cglib 字节码的方式把代理类反射出来，而代理类在内存中就会对某种类型的类进行调用，invocationHandler（Jdk）或者 MethodInterceptor（cglib）

5. 怎么理解面向切面编程的切面?

答:

切面指的是一类功能，比如事务，缓存，日志等。切面的作用是在目标对象方法执行前或者后执行切面方法

6. 什么是 IOC 容器？

答:

Spring IOC 负责创建对象、管理对象(通过依赖注入)、整合对象、配置对象以及管理这些对象的生命周期。

Ioc 是把对象的控制权较给框架或容器，容器中存储了众多我们需要的对象，然后我们就无需再手动的在代码中创建对象。需要什么对象就直接告诉容器我们需要什么对象，容器


微信搜一搜
享学课堂online

会把对象根据一定的方式注入到我们的代码中。注入的过程被称为 DI。有时候需要动态的指定我们需要什么对象，这个时候要让容器在众多对象中去寻找，容器寻找需要对象的过程，称为 DL(Dependency Lookup, 依赖查找)。按照上面的理解，那么 IOC 包含了 DI 与 DL，并且多了对象注册的过程。

7. 为什么需要代理模式?

答：

代理其实是对 开闭原则和里氏代换原则的一种实现，它强调在不修改老代码的基础上扩展功能，代理模式大大提高了代码的灵活性，同时也使代码的可读性变差，同时动态代理也加大了内存溢出的风险

8. 讲讲静态代理模式的优点及其瓶颈?

答：

静态代理一种傻瓜式的对目标类的增强，其核心就两点，1、跟目标类实现同一接口 2、持有目标类的引用。 如果需要对目标类方法进行增强，就只要调用代理类的方法，在方法里面写增强功能然后调用目标类方法即可。优点就是代码直观且执行速度快，缺点就是不够灵活，代码量比较大，每一种类型的目标类都需要单独写一个代理类。

9. 讲解 Spring 框架中基于 Schema 的 AOP 实现原理?

答：

这个是基于 xml 配置方式的 aop

```
<aop:config proxy-target-class="true">
<aop:pointcut expression="execution(public void org.aop.Object.Student.display())" id="pointcut"/> <!-- 切点的声明定义-->
 <aop:aspect ref="advice"> <!-- 关联通知类-->
<aop:before method="before" pointcut-ref="pointcut"/> <!-- 前置通知-->
<aop:after-returning method="after" pointcut-ref="pointcut" /><!-- 后置通知-->
<aop:after-throwing method="exception" pointcut-ref="pointcut" throwing="x"/><!-- 异常通知-->
<aop:around method="around" pointcut-ref="pointcut"/><!-- 环绕通知-->
 </aop:aspect>
 </aop:config>
```

AOP 实现流程


微信搜一搜
享学课堂online

1、aop:config 自定义标签解析

2、自定义标签解析时封装对应的 aop 入口类，类的类型就是 BeanPostProcessor 接口类型

3、Bean 实现化过程中会执行到 aop 入口类中

4、在 aop 入口类中，判断当前正在实例化的类是否在 pointcut 中，pointcut 可以理解为一个模糊匹配，是一个 joinpoint 的集合

5、如果当前正在实例化的类在 pointcut 中，则返回该 bean 的代理，同时把所有配置的 advice 封装成 MethodInterceptor 对象加入到容器中，封装成一个过滤器链

代理对象调用，jdk 调到 invocationHandler 中，cglib 调到 MethodInterceptor 的 callback 类中，然后在 invoke 方法中执行过滤器链。

10. 讲解 Spring 框架中如何基于 AOP 实现的事务管理?

答：

事务管理，是一个切面。在 aop 环节中，其他环节都一样，事务管理就是有自己单独的 advice，有单独的通知，是 TransactionInterceptor，它一样的会在拦截器链中被执行到，这个 TransactionInterceptor 拦截器类是通过解析<tx:advice>自定义标签得到的。

11. Spring 在 Bean 创建过程中是如何解决循环依赖的?

答：

循环依赖只会存在在单例实例中，多例循环依赖直接报错。

A 类实例化后，把实例放 map 容器中，A 类中有一个 B 类属性，A 类实例化要进行 IOC 依赖注入，这时候 B 类需要实例化，B 类实例化跟 A 类一样，实例化后方 map 容器中。B 类中有一个 A 类属性，接着 B 类的 IOC 过程，又去实例化 A 类，这时候实例化 A 类过程中从 map 容器发现 A 类已经在容器中了，就直接返回了 A 的实例，依赖注入到 B 类中 A 属性中，B 类 IOC 完成后，B 实例化就完全完成了，就返回给 A 类的 IOC 过程。这就是循环依赖的解决。

12. spring 中 Bean 是如何管理的？

答：

在 Spring 框架中，一旦把一个 bean 纳入到 Spring IoC 容器之中，这个 bean 的生命周


微信搜一搜
享学课堂online

期就会交由容器进行管理，一般担当管理者角色的是 BeanFactory 或 ApplicationContext。认识一下 Bean 的生命周期活动，对更好的利用它有很大的帮助。

概括来说主要有四个阶段：实例化，初始化，使用，销毁。

13. 请具体描述 IOC 容器对 Bean 的生命周期控制流程:

通过构造器或工厂方法创建 Bean 实例

. 为 Bean 的属性设置值和对其他 Bean 的引用

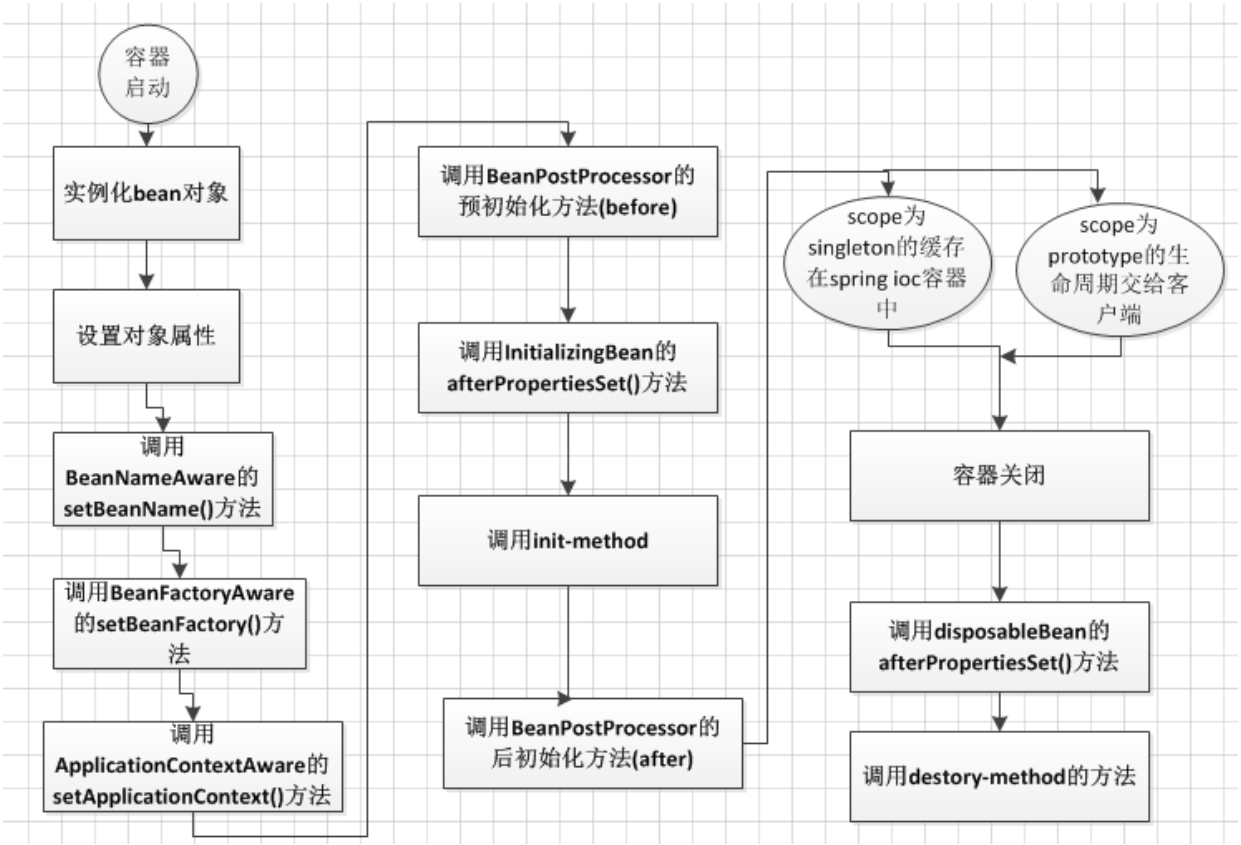
. 将 Bean 实例传递给 Bean 后置处理器的 postProcessBeforeInitialization 方法

. 调用 Bean 的初始化方法(init-method)

. 将 Bean 实例传递给 Bean 后置处理器的 postProcessAfterInitialization 方法

. Bean 可以使用了

. 当容器关闭时, 调用 Bean 的销毁方法(destroy-method)



14. BeanFactory 和 ApplicationContext 有什么区别?

1)BeanFactory 是 Spring 里面最底层的接口，包含了各种 Bean 的定义，读取 bean

微信搜一搜
享学课堂online

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。

2)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。ApplicationContext 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。

3)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。

4)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。

19. 谈谈 Spring Bean 创建过程中的设计模式?
策略模式、代理模式、适配器模式

15. Spring MVC 运行流程

答：

第一步：发起请求到前端控制器(DispatcherServlet)

第二步：前端控制器请求 HandlerMapping 查找 Handler（ 可以根据 xml 配置、注解进行查找）

第三步：处理器映射器 HandlerMapping 向前端控制器返回 Handler

第四步：前端控制器调用处理器适配器去执行 Handler

第五步：处理器适配器去执行 Handler

第六步：Handler 执行完成给适配器返回 ModelAndView

第七步：处理器适配器向前端控制器返回 ModelAndView（ModelAndView 是 springmvc 框架的一个底层对象，包括 Model 和 view）

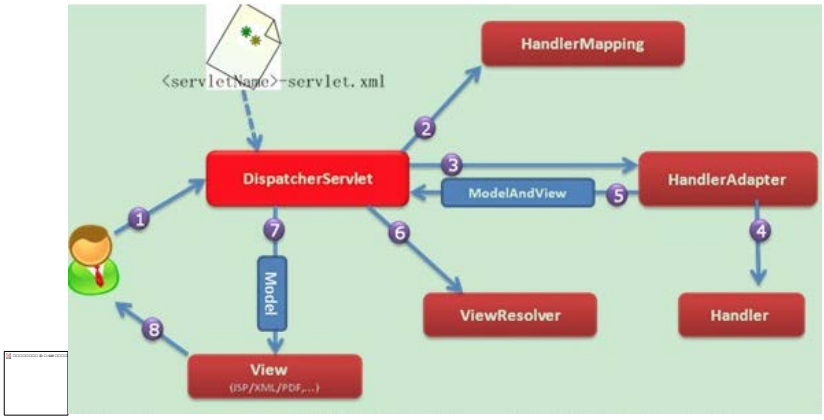
第八步：前端控制器请求视图解析器去进行视图解析（根据逻辑视图名解析成真正的视图(jsp)）

第九步：视图解析器向前端控制器返回 View


微信搜一搜
享学课堂online

第十步：前端控制器进行视图渲染（ 视图渲染将模型数据(在 ModelAndView 对象中)填充到 request 域）

第十一步：前端控制器向用户响应结果



16. Spring 框架中的单例 bean 是线程安全的吗?

一般的 Web 应用划分为展现层、服务层和持久层三个层次，在不同的层中编写对应的逻辑，下层通过接口向上层开放功能调用。在一般情况下，从接收请求到返回响应所经过的所有程序调用都同属于一个线程

ThreadLocal 是解决线程安全问题一个很好的思路，它通过为每个线程提供一个独立的变量副本解决了变量并发访问的冲突问题。在很多情况下，ThreadLocal 比直接使用 synchronized 同步机制解决线程安全问题更简单，更方便，且结果程序拥有更高的并发性。

线程安全问题都是由全局变量及静态变量引起的。

无状态就是一次操作，不能保存数据。无状态对象(Stateless Bean)，就是没有实例变量的对象 .不能保存数据，是不变类，是线程安全的。

当然也可以通过加 sync 锁的方法来解决线程安全，这种以时间换空间的场景在高并发场景下显然是不实际的。

17. 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 Springioc 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 HTTPSession 中，一个 bean 定义对应一个实例。该作用域仅在基于 web 的 SpringApplicationContext 情形下有效。
- global-session：在一个全局的 HTTPSession 中，一个 bean 定义对应一个实例。该作用域仅在基于 web 的 SpringApplicationContext 情形下有效。

微信搜一搜
享学课堂online

缺省的 Springbean 的作用域是 Singleton.

18.解释 Spring 框架中 bean 的生命周期

Spring 容器从 XML 文件中读取 bean 的定义，并实例化 bean。
☐ Spring 根据 bean 的定义填充所有的属性。
☐ 如果 bean 实现了 BeanNameAware 接口，Spring 传递 bean 的 ID 到 setBeanName 方法。
☐ 如果 Bean 实现了 BeanFactoryAware 接口，Spring 传递 beanfactory 给 setBeanFactory 方法。
☐ 如果有任何与 bean 相关联的 BeanPostProcessors，Spring 会在 postProcesserBeforeInitialization()方法内调用它们。
☐ 如果 bean 实现 IntializingBean 了，调用它的 afterPropertySet 方法，如果 bean 声明了初始化方法，调用此初始化方法。
☐ 如果有 BeanPostProcessors 和 bean 关联，这些 bean 的 postProcessAfterInitialization()方法将被调用。
☐ 如果 bean 实现了 DisposableBean，它将调用 destroy()方法。

19.哪些是重要的 bean 生命周期方法？你能重载它们吗？

有两个重要的 bean 生命周期方法，第一个是 setup，它是在容器加载 bean 的时候被调用。第二个方法是 teardown 它是在容器卸载类的时候被调用。Thebean 标签有两个重要的属性（init-method 和 destroy-method）。用它们你可以自己定制初始化和注销方法。它们也有相应的注解（@PostConstruct 和@PreDestroy）。


微信搜一搜
享学课堂online