**Spring中bean的作用域和生命周期**

1. singleton
   1. springIoc容器中只会存在一个共享的bean实例，无论多少个bean引用它，始终指向同一个对象，缺省值
2. prototype
   1. 每次通过spring容器获取prototype定义的bean时，容器都会创建一个新的 bean实例，每个bean实例都有自己的属性和状态
3. Request
   1. 在一次http请求中，容器会返回该bean的同一实例，而对不同的http请求则会产生新的bean，该bean只对当前httprequest有效
4. Session
   1. 同上httpSession
5. Global Session
   1. 这是对全局的session来说，仅在使用portlet context时有效

**生命周期**

1. 实例化一个bean也就是new
2. 按照spring上下文对实例化的bean进行配置，也就是ioc注入
   1. 如果这个bean implements BeanNameAware interface 会调用它实现的setBeanName(String beanid) 此处传递的是spring配置文件中bean的id
   2. 如果这个bean implements BeanFactoryAware interface 会调用它实现的setBeanFactory()传递的是spring工厂本身（还可以用这个方法获取其他的bean）
   3. 如果这个bean关联了BeanPostProcessor接口，将会调用postProccessBeforeInittializatio(Object obj,String s),BeanProccessor经常被用作bean内容的修改，由于这个是在bean初始化结束时调用After方法，也可用于内存或缓存技术
   4. 如果这个bean在spring配置文件中配置了init-method属性，会自动调用其配置的初始化方法
   5. 如果这个bean关联了beanpostProccessor接口，将会调用posAfterInitialization(Object obj)
3. 当bean不再需要时，会经过清理阶段，
   1. 如果bean实现了disposablebean接口，会调用其实现的destroy方法
   2. 如果这个bean的spring配置中配置了destroy-method属性，会自动调用其配置的销毁方法

**对spring依赖注入的两种方式的认识**

1. 构造方法和设值注入
   1. 设值注入和传统的JavaBean的写法更相似，通过setter方法设定依赖关系显得更直观
   2. 对于复杂的依赖关系，如果采用构造注入，会导致构造器过于臃肿，Spring在创建bean实例时，需要同时实例化其依赖的全部实例，因而会产生浪费，使用设值注入，避免这个问题，在某些属性可选的情况下，也是鼓励使用设值注入
   3. 构造注入可以在构造器中决定依赖关系的注入顺序。优先依赖的优先注入
   4. 对于依赖关系无需变化的bean，构造注入更有用处，因为没有setter方法，所有的依赖关系全部在构造器内设定，只有组件的创建者才能改变组件的依赖关系，对组件的调用者而言，组件内部的依赖关系完全透明，更符合高内聚的原则
   5. 设值注入不会重写构造方法的值，如果对同一变量同时使用了构造方法注入有使用了设值方法，构造方法不会覆盖设值的注入
   6. 建议采用设值注入为主，构造注入为辅的注入策略，
   7. 对于依赖关系无需变化的注入，尽量采用构造注入，而其他依赖则set注入

**Spring框架用到了哪些设计模式**

1. 代理模式：在AOP和remoting中备用的比较多
2. 单例模式：在spring配置文件中定义的bean默认为单例模式
3. 模板方法模式：用来解决代码重复的问题
4. 前端控制模式：Spring提供了DispatcherServlet来对请求进行分发
5. 依赖注入模式：贯穿于BeanFactory/ApplicatonContext接口的核心理念
6. 工厂模式：beanFactory用来创建对象的实例

BeanFactory和Applicationcontext的区别

1. BeanFactory和AppliationContext都是接口，并且ApplicationContext是BeanFactory的子接口
2. BeanFactory是Spring中最底层额接口，提供了最简单的容器的功能，提供了实例化对象和拿对象的功能，
3. ApplicationContext是Spring的一个更高级的容器，提供更多有用的功能
   1. 国际化、消息发送、响应机制、统一加载资源、强大的事件机制，对web应用的支持等
4. 加载的区别：
   1. BeanFactory采用延迟加载注入bean
   2. ApplicationContext在IOC启动时就一次性创建所有的bean，好处是可以马上发现spring配置文件中的错误，坏处是造成浪费

**Spring的核心模块**

1. Spring Core[核心容器]：提供Spring的基本功能，核心容器的核心功能是用IOC容器管理类的依赖关系
2. Spring AOP[面向切面]：Spring的AOP模块提供了面向切面编程的支持，Spring AOP采用的是纯java实现，采用基于代理的AOP实现方案，AOP代理由IOC容器负责生成管理，依赖关系也一并由IOC容器管理
3. Spring ORM[对象实体映射]：提供了与多个第三方持久层框架的良好整合
4. Spring DAO[持久层模块]：Spring进一步简化DAO开发步骤，能以一致的方式使用数据库访问技术，用统一的方式调用事务管理，避免具体的实现侵入业务逻辑层的代码中
5. Spring Context[应用上下文]：他是一个配置文件，为Spring提供上下文信息，提供了框架式的对象访问方法
6. Spring Web[web模块]：提供了基础的针对web开发的集成特性
7. Spring MVC[MVC模块]：提供了web应用的MVC实现，Spring的MVC框架并不是仅仅提供一种传统的实现，他提供了一种清晰的分离模型