说一下 spring 中 Bean 的作用域

singleton：

Spring IoC 容器中只会存在一个共享的 Bean 实例，无论有多少个 Bean 引用它，始终指向同一对象。Singleton 作用域是 Spring 中的缺省作用域。

prototype：

每次通过 Spring 容器获取 prototype 定义的 bean 时，容器都将创建一个新的 Bean 实例，每个 Bean 实例都有自己的属性和状态，而 singleton 全局只有一个对象。

request：

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，而且该 bean 仅在当前 Http Request 内有效。

session：

在一次 Http Session 中，容器会返回该 Bean 的同一实例。而对不同的 Session 请求则会创建新的实例，该 bean 实例仅在当前 Session 内有效。

global Session：

在一个全局的 Http Session 中，容器会返回该 Bean 的同一个实例，仅在使用 portlet context 时有效。

说一下 spring 中 Bean 的生命周期

- 实例化一个 Bean，也就是我们通常说的 new。
- 按照 Spring 上下文对实例化的 Bean 进行配置，也就是 IOC 注入。
- 如果这个 Bean 实现了 BeanNameAware 接口，会调用它实现的 setBeanName(String beanId)方法，此处传递的是 Spring 配置文件中 Bean 的 ID。
- 如果这个 Bean 实现了 BeanFactoryAware 接口，会调用它实现的 setBeanFactory()，传递的是 Spring 工厂本身（可以用这个方法获取到其他 Bean）。
- 如果这个 Bean 实现了 ApplicationContextAware 接口，会调用 setApplicationContext(ApplicationContext)方法，传入 Spring 上下文。
- 如果这个 Bean 关联了 BeanPostProcessor 接口，将会调用 postProcessBeforeInitialization(Object obj, String s)方法，

BeanPostProcessor 经常被用作是 Bean 内容的更改，并且由于这个是在 Bean 初始化结束时调用 After 方法，也可用于内存或缓存技术。

- 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。
- 如果这个 Bean 关联了 BeanPostProcessor 接口，将会调用 postAfterInitialization(Object obj, String s)方法。
- 当 Bean 不再需要时，会经过清理阶段，如果 Bean 实现了 DisposableBean 接口，会调用其实现的 destroy 方法。
- 最后，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 中依赖注入两种方式的认识

两种注入方式为：构造方法注入和设值注入

1. 设值注入与传统的 JavaBean 的写法更相似，程序员更容易理解、接受，通过 setter 方式设定依赖关系显得更加直观、明显;
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，难以阅读。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。而使用设置注入，则避免这下问题;
3. 在某些属性可选的情况下，多参数的构造器更加笨拙，官方更鼓励使用设值注入。
4. 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。
5. 对于依赖关系无须变化的 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，因此，不用担心后续代码对依赖关系的破坏。
6. 构造注入使依赖关系只能在构造器中设定，则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。
7. 设值注入不会重写构造方法的值。如果我们对同一个变量同时使用了构造方法注入又使用了设置方法注入的话，那么构造方法将不能覆盖由设值方法注入的值。
8. 建议采用以设值注入为主，构造注入为辅的注入策略。对于依赖关系无须变化的注入，尽量采用构造注入;而其他的依赖关系的注入，则考虑采用 set 注入。

Spring 框架中都用到了哪些设计模式？

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

BeanFactory 和 ApplicationContext 的区别

BeanFactory 和 ApplicationContext 都是接口，并且 ApplicationContext 是 BeanFactory 的子接口。

BeanFactory 是 Spring 中最底层的接口，提供了最简单的容器的功能，只提供了实例化对象和拿对象的功能。而 ApplicationContext 是 Spring 的一个更高级的容器，提供了更多的有用的功能。

ApplicationContext 提供的额外的功能：国际化的功能、消息发送、响应机制、统一加载资源的功能、强大的事件机制、对 Web 应用的支持等等。

加载方式的区别：BeanFactory 采用的是延迟加载的形式来注入 Bean；ApplicationContext 则相反的，它是在 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。