spring

介绍 spring 相关技术

- spring 事务传播机制
- bean 是线程安全的吗
- spring 设计模式

spring ??????

spring的事务传播机制是什么?

多个事务方法相互调用时，事务如何在这些方法之间进行传播,spring中提供了7中不同的传播特性，来保证事务的正常执行：

REQUIRED：默认的传播特性，如果当前没有事务，则新建一个事务，如果当前存在事务，则加入这个事务

SUPPORTS：当前存在事务，则加入当前事务，如果当前没有事务，则以非事务的方式执行

MANDATORY：当前存在事务，则加入当前事务，如果当前事务不存在，则抛出异常

REQUIRED_NEW：创建一个新事务，如果存在当前事务，则挂起改事务

NOT_SUPPORTED：以非事务方式执行，如果存在当前事务，则挂起当前事务

NEVER：不使用事务，如果当前事务存在，则抛出异常

NESTED：如果当前事务存在，则在嵌套事务中执行，否则REQUIRED的操作一样

NESTED和REQUIRED_NEW的区别：

REQUIRED_NEW是新建一个事务并且新开始的这个事务与原有事务无关，而NESTED则是当前存在事务时会开启一个嵌套事务，在NESTED情况下，父事务回滚时，子事务也会回滚，而REQUIRED_NEW情况下，原有事务回滚，不会影响新开启的事务

NESTED和REQUIRED的区别：

REQUIRED情况下，调用方存在事务时，则被调用方和调用方使用同一个事务，那么被调用方出现异常时，由于共用一个事务，所以无论是否catch异常，事务都会回滚，而在NESTED情况下，被调用方发生异常时，调用方可以catch其异常，这样只有子事务回滚，父事务不会回滚

NESTED：如果当前事务存在，则在嵌套事务中执行，否则REQUIRED的操作一样

NESTED和REQUIRED_NEW的区别：

REQUIRED_NEW是新建一个事务并且新开始的这个事务与原有事务无关，而NESTED则是当前存在事务时会开启一个嵌套事务，在NESTED情况下，父事务回滚时，子事务也会回滚，而REQUIRED_NEW情况下，原有事务回滚，不会影响新开启的事务

NESTED和REQUIRED的区别：

REQUIRED情况下，调用方存在事务时，则被调用方和调用方使用同一个事务，那么被调用方出现异常时，由于共用一个事务，所以无论是否catch异常，事务都会回滚，而在NESTED情况下，被调用方发生异常时，调用方可以catch其异常，这样只有子事务回滚，父事务不会回滚。

bean ???????

Spring框架中的单例Bean是线程安全的么?

Spring中的Bean对象默认是单例的，框架并没有对bean进行多线程的封装处理

如果Bean是有状态的，那么就需要开发人员自己来保证线程安全的保证，最简单的办法就是改变bean的作用域把singleton改成prototype，这样每次请求bean对象就相当于是创建新的对象来保证线程的安全

有状态就是由数据存储的功能

无状态就是不会存储数据，你想一下，我们的controller，service和dao本身并不是线程安全的，只是调用里面的方法，而且多线程调用一个实例的方法，会在内存中复制遍历，这是自己线程的工作内存，是最安全的。

因此在进行使用的时候，不要在bean中声明任何有状态的实例变量或者类变量，如果必须如此，也推荐大家使用ThreadLocal把变量变成线程私有，如果bean的实例变量或者类变量需要在多个线程之间共享，那么就只能使用synchronized，lock，cas等这些实现线程同步的方法了。

spring ????

spring框架中使用了哪些设计模式及应用场景

1.工厂模式，在各种BeanFactory以及ApplicationContext创建中都用到了

2.模版模式，在各种BeanFactory以及ApplicationContext实现中也都用到了

3.代理模式，Spring AOP 利用了 AspectJ AOP实现的! AspectJ AOP 的底层用了动态代理

4.策略模式，加载资源文件的方式，使用了不同的方法，比如：ClassPathResourece，FileSystemResource，ServletContextResource，UrlResource但他们都有共同的借口Resource；在Aop的实现中，采用了两种不同的方式，JDK动态代理和CGLIB代理

5.单例模式，比如在创建bean的时候。

6.观察者模式，spring中的ApplicationEvent，ApplicationListener,ApplicationEventPublisher

7.适配器模式，MethodBeforeAdviceAdapter,ThrowsAdviceAdapter,AfterReturningAdapter

8.装饰者模式，源码中类型带Wrapper或者Decorator的都是