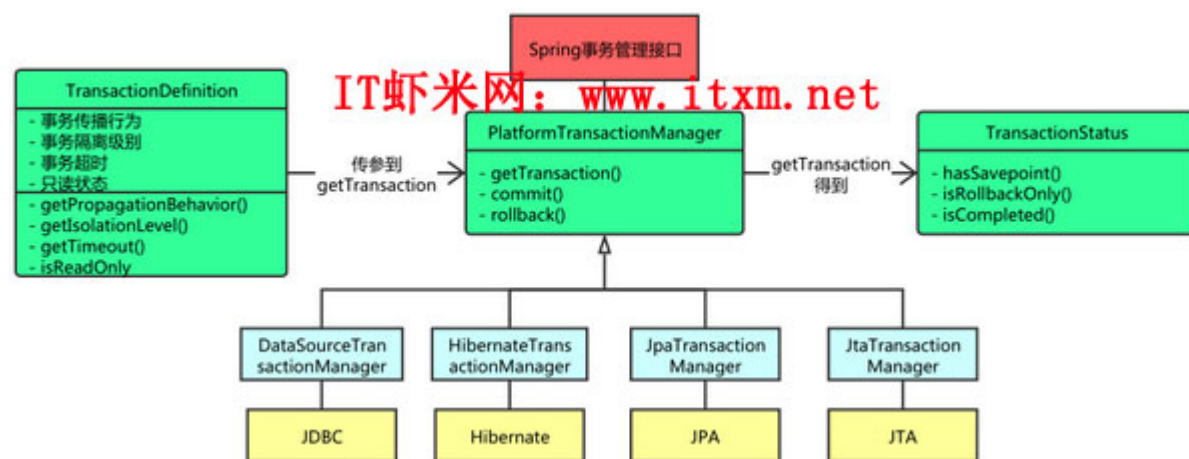


## 深入理解 Spring 事务原理

转载

2017年11月11日 22:38:20

83



### 1、Spring中事务处理的作用：

Spring事务处理，是将事务处理的工作统一起来，并为事务处理提供通用的支持。

### 2、工作原理及实现

#### a、划分处理单元——IOC

由于spring解决的问题是对单个数据库进行局部事务处理的，具体的实现首相用spring中的IOC划分了事务处理单元。并且将对事务的各种配置放到了ioc容器中（设置事务管理器，设置事务的传播特性及隔离机制）。

#### b、AOP拦截需要进行事务处理的类

Spring事务处理模块是通过AOP功能来实现声明式事务处理的，具体操作（比如事务实行的配置和读取，事务对象的抽象），用TransactionProxyFactoryBean接口来使用AOP功能，生成proxy代理对象，通过TransactionInterceptor完成对代理方法的拦截，将事务处理的功能编织到拦截的方法中。读取ioc容器事务配置属性，转化为spring事务处理需要的内部数据结构（TransactionAttributeSourceAdvisor），转化为TransactionAttribute表示的数据对象。

#### c、对事物处理实现（事务的生成、提交、回滚、挂起）

spring委托给具体的事务处理器实现。实现了一个抽象和适配。适配的具体事务处理器：DataSource数据源支持、hibernate数据源事务处理支持、JDO数据源事务处理支持，JPA、JTA数据源事务处理支持。这些支持都是通过设计PlatformTransactionManager、AbstractPlatformTransactionManager一系列事务处理的支持。为常用数据源支持提供了一系列的TransactionManager。

#### d、结合

PlatformTransactionManager实现了TransactionInterception接口，让其与TransactionProxyFactoryBean结合起

来，形成一个Spring声明式事务处理的设计体系。

### 3、应用场景

支持不同数据源，在底层进行封装，可以做到事务即开即用，这样的好处是：即使有其他的数据源事务处理需要，Spring也提供了一种一致的方式。

#### 一、事务的基本原理

Spring事务 的本质其实就是数据库对事务的支持，没有数据库的事务支持，spring是无法提供事务功能的。对于纯JDBC操作数据库，想要用到事务，可以按照以下步骤进行：

获取连接 `Connection con = DriverManager.getConnection()`

开启事务`con.setAutoCommit(true/false);`

执行CRUD

提交事务/回滚事务 `con.commit()` / `con.rollback();`

关闭连接 `conn.close();`

使用Spring的事务管理功能后，我们可以不再写步骤 2 和 4 的代码，而是由Spring 自动完成。那么Spring是如何在我们书写的 CRUD 之前和之后开启事务和关闭事务的呢？解决这个问题，也就可以从整体上理解Spring的事务管理实现原理了。下面简单地介绍下，注解方式为例子

配置文件开启注解驱动，在相关的类和方法上通过注解@Transactional标识。

spring 在启动的时候会去解析生成相关的bean，这时候会查看拥有相关注解的类和方法，并且为这些类和方法生成代理，并根据@Transaction的相关参数进行相关配置注入，这样就在代理中为我们把相关的事务处理掉了（开启正常提交事务，异常回滚事务）。

真正的数据库层的事务提交和回滚是通过binlog或者redo log实现的。

#### 二、Spring 事务的传播属性

所谓spring事务的传播属性，就是定义在存在多个事务同时存在的时候，spring应该如何处理这些事务的行为。这些属性在TransactionDefinition中定义，具体常量的解释见下表：

常量名称      常量解释

PROPAGATION\_REQUIRED      支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择，也是Spring 默认的事务的传播。

PROPAGATION\_REQUIRES\_NEW      新建事务，如果当前存在事务，把当前事务挂起。新建的事务将和被挂起的事务没有任何关系，是两个独立的事务，外层事务失败回滚之后，不能回滚内层事务执行的结果，内层事务失败抛出异常，外层事务捕获，也可以不处理回滚操作

PROPAGATION\_SUPPORTS      支持当前事务，如果当前没有事务，就以非事务方式执行。

PROPAGATION\_MANDATORY      支持当前事务，如果当前没有事务，就抛出异常。

PROPAGATION\_NOT\_SUPPORTED      以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

PROPAGATION\_NEVER      以非事务方式执行，如果当前存在事务，则抛出异常。

PROPAGATION\_NESTED

如果一个活动的事务存在，则运行在一个嵌套的事务中。如果没有活动事务，则按REQUIRED属性执行。它使用了一个单

独的事务，这个事务拥有多个可以回滚的保存点。内部事务的回滚不会对外部事务造成影响。它只对DataSourceTransactionManager事务管理器起效。

### 三、数据库隔离级别

隔离级别    隔离级别的值    导致的问题

Read-Uncommitted    0    导致脏读

Read-Committed    1    避免脏读，允许不可重复读和幻读

Repeatable-Read    2    避免脏读，不可重复读，允许幻读

Serializable    3    串行化读，事务只能一个一个执行，避免了脏读、不可重复读、幻读。执行效率慢，使用时慎重

脏读：一事务对数据进行了增删改，但未提交，另一事务可以读取到未提交的数据。如果第一个事务这时候回滚了，那么第二个事务就读到了脏数据。

不可重复读：一个事务中发生了两次读操作，第一次读操作和第二次操作之间，另外一个事务对数据进行了修改，这时候两次读取的数据是不一致的。

幻读：第一个事务对一定范围的数据进行批量修改，第二个事务在这个范围增加一条数据，这时候第一个事务就会丢失对新增数据的修改。

数据库事务的隔离级别有4个，由低到高依次为Read uncommitted、Read committed、Repeatable read、Serializable，这四个级别可以逐个解决脏读、不可重复读、幻读这几类问题。

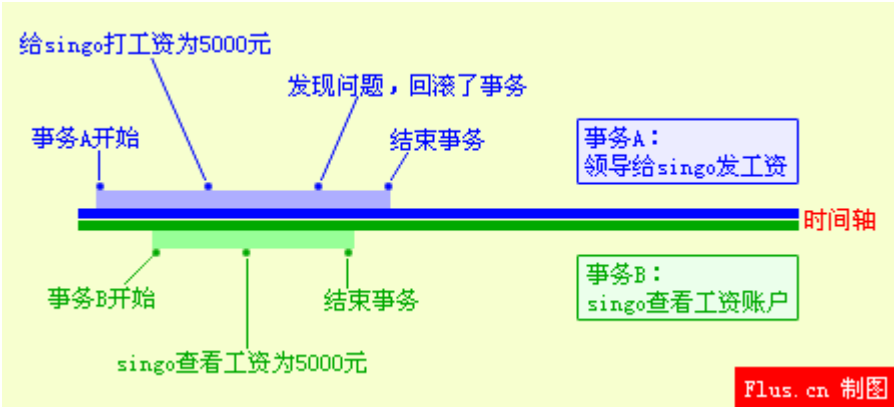
√: 可能出现    ×: 不会出现

	脏读	不可重复读	幻读
Read uncommitted	√	√	√
Read committed	×	√	√
Repeatable read	×	×	√
Serializable	×	×	×

注意：我们讨论隔离级别的场景，主要是在多个事务并发的情况下，因此，接下来的讲解都围绕事务并发。

#### Read uncommitted 读未提交

公司发工资了，领导把5000元打到singo的账号上，但是该事务并未提交，而singo正好去查看账户，发现工资已经到账，是5000元整，非常高兴。可是不幸的是，领导发现发给singo的工资金额不对，是2000元，于是迅速回滚了事务，修改金额后，将事务提交，最后singo实际的工资只有2000元，singo空欢喜一场。



出现上述情况，即我们所说的脏读，两个并发的任务，“事务A：领导给singo发工资”、“事务B：singo查询工资账户”，事务B读取了事务A尚未提交的数据。

当隔离级别设置为Read uncommitted时，就可能出现脏读，如何避免脏读，请看下一个隔离级别。

#### Read committed 读提交

singo拿着工资卡去消费，系统读取到卡里确实有2000元，而此时她的老婆也正好在网上转账，把singo工资卡的2000元转到另一账户，并在singo之前提交了事务，当singo扣款时，系统检查到singo的工资卡已经没钱，扣款失败，singo十分纳闷，明明卡里有钱，为何.....

出现上述情况，即我们所说的不可重复读，两个并发的任务，“事务A：singo消费”、“事务B：singo的老婆网上转账”，事务A事先读取了数据，事务B紧接了更新了数据，并提交了事务，而事务A再次读取该数据时，数据已经发生了改变。

当隔离级别设置为Read committed时，避免了脏读，但是可能会造成不可重复读。

大多数数据库的默认级别就是Read committed，比如Sql Server，Oracle。如何解决不可重复读这一问题，请看下一个隔离级别。

#### Repeatable read 重复读

当隔离级别设置为Repeatable read时，可以避免不可重复读。当singo拿着工资卡去消费时，一旦系统开始读取工资卡信息（即事务开始），singo的老婆就不可能对该记录进行修改，也就是singo的老婆不能在此时转账。

虽然Repeatable read避免了不可重复读，但还有可能出现幻读。

singo的老婆工作在银行部门，她时常通过银行内部系统查看singo的信用卡消费记录。有一天，她正在查询到singo当月信用卡的总消费金额（select sum(amount) from transaction where month = 本月）为80元，而singo此时正好在外面胡吃海塞后在收银台买单，消费1000元，即新增了一条1000元的消费记录（insert transaction ...），并提交了事务，随后singo的老婆将singo当月信用卡消费的明细打印到A4纸上，却发现消费总额为1080元，singo的老婆很诧异，以为出现了幻觉，幻读就这样产生了。

注：Mysql的默认隔离级别就是Repeatable read。

#### Serializable 序列化

Serializable是最高的事务隔离级别，同时代价也花费最高，性能很低，一般很少使用，在该级别下，事务顺序执行，不仅可以避免脏读、不可重复读，还避免了幻像读。

总结：

隔离级别越高，越能保证数据的完整性和一致性，但是对并发性能的影响也越大。

大多数的数据库默认隔离级别为 Read Committed，比如 SqlServer、Oracle

少数数据库默认隔离级别为：Repeatable Read 比如：MySQL InnoDB

#### 四、Spring中的隔离级别

常量     解释

ISOLATION\_DEFAULT     这是个 PlatfromTransactionManager 默认的隔离级别，使用数据库默认的事务隔离级别。

另外四个与 JDBC 的隔离级别相对应。

ISOLATION\_READ\_UNCOMMITTED     这是事务最低的隔离级别，它充许另外一个事务可以看到这个事务未提交的数据。这种隔离级别会产生脏读，不可重复读和幻像读。

ISOLATION\_READ\_COMMITTED     保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数据。

ISOLATION\_REPEATABLE\_READ     这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。

ISOLATION\_SERIALIZABLE     这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行。

#### 五、事务的嵌套

通过上面的理论知识的铺垫，我们大致知道了数据库事务和spring事务的一些属性和特点，接下来我们通过分析一些嵌套事务的场景，来深入理解spring事务传播的机制。

假设外层事务 Service A 的 Method A() 调用 内层Service B 的 Method B()

PROPAGATION\_REQUIRED(spring 默认)

如果ServiceB.methodB() 的事务级别定义为 PROPAGATION\_REQUIRED，那么执行 ServiceA.methodA() 的时候 spring 已经起了事务，这时调用 ServiceB.methodB()，ServiceB.methodB() 看到自己已经运行在 ServiceA.methodA() 的事务内部，就不再起新的事务。

假如 ServiceB.methodB() 运行的时候发现自己没有在事务中，他就会为自己分配一个事务。

这样，在 ServiceA.methodA() 或者在 ServiceB.methodB() 内的任何地方出现异常，事务都会被回滚。

PROPAGATION\_REQUIRES\_NEW

比如我们设计 ServiceA.methodA() 的事务级别为 PROPAGATION\_REQUIRED，ServiceB.methodB() 的事务级别为 PROPAGATION\_REQUIRES\_NEW。

那么当执行到 ServiceB.methodB() 的时候，ServiceA.methodA() 所在的事务就会挂起，ServiceB.methodB() 会起一个新的事务，等待 ServiceB.methodB() 的事务完成以后，它才继续执行。

他与 PROPAGATION\_REQUIRED 的事务区别在于事务的回滚程度了。因为 ServiceB.methodB() 是新起一个事务，那么就是存在两个不同的事务。如果 ServiceB.methodB() 已经提交，那么 ServiceA.methodA() 失败回滚，ServiceB.methodB() 是不会回滚的。如果 ServiceB.methodB() 失败回滚，如果他抛出的异常被 ServiceA.methodA() 捕获，ServiceA.methodA() 事务仍然可能提交(主要看B抛出的异常是不是A会回滚的异常)。

PROPAGATION\_SUPPORTS

假设ServiceB.methodB() 的事务级别为 PROPAGATION\_SUPPORTS，那么当执行到ServiceB.methodB()时，如果发现ServiceA.methodA()已经开启了一个事务，则加入当前的事务，如果发现ServiceA.methodA()没有开启事务，则自己也不开启事务。这种时候，内部方法的事务性完全依赖于最外层的事务。

PROPAGATION\_NESTED

现在的情况就变得比较复杂了，ServiceB.methodB() 的事务属性被配置为 PROPAGATION\_NESTED, 此时两者之间又

将如何协作呢？ ServiceB#methodB 如果 rollback, 那么内部事务(即 ServiceB#methodB) 将回滚到它执行前的 SavePoint 而外部事务(即 ServiceA#methodA) 可以有以下两种处理方式:

a、捕获异常，执行异常分支逻辑

```
void methodA() {
    try {
        ServiceB.methodB();
    } catch (SomeException) {
        // 执行其他业务, 如 ServiceC.methodC();
    }
}
```

这种方式也是嵌套事务最有价值的地方，它起到了分支执行的效果，如果 ServiceB.methodB 失败，那么执行 ServiceC.methodC(), 而 ServiceB.methodB 已经回滚到它执行之前的 SavePoint, 所以不会产生脏数据(相当于此方法从未执行过)，这种特性可以用在某些特殊的业务中，而 PROPAGATION\_REQUIRED 和 PROPAGATION\_REQUIRES\_NEW 都没有办法做到这一点。

b、 外部事务回滚/提交 代码不做任何修改，那么如果内部事务(ServiceB#methodB) rollback, 那么首先 ServiceB.methodB 回滚到它执行之前的 SavePoint(在任何情况下都会如此), 外部事务(即 ServiceA#methodA) 将根据具体的配置决定自己是 commit 还是 rollback

另外三种事务传播属性基本用不到，在此不做分析。

<tx:advice/> 和 <tx:attributes/> 标签里的 <tx:method/> 各种属性设置总结如下：

表 <tx:method/> 有关的设置

属性	是否需要？	默认值	描述
name	是		与事务属性关联的方法名。通配符（*）可以用来指定一批关联到相同的事务属性的方法。如：'get*'、'handle*'、'on*Event'等等。
propagation	不	REQUIRED	事务传播行为
isolation	不	DEFAULT	事务隔离级别
timeout	不	-1	事务超时的时间（以秒为单位）
read-only	不	false	事务是否只读？
rollback-for	不		将被触发进行回滚的 Exception(s)；以逗号分开。 如：'com.foo.MyBusinessException,ServletException'
no-rollback-for	不		不被触发进行回滚的 Exception(s)；以逗号分开。 如：'com.foo.MyBusinessException'

## 六、总结

对于项目中需要使用到事务的地方，我建议开发者还是使用spring的TransactionCallback接口来实现事务，不要盲目使

用spring事务注解，如果一定要使用注解，那么一定要对spring事务的传播机制和隔离级别有个详细的了解，否则很可能发生意想不到的效果。



目前您尚未登录，请 [登录](#) 或 [注册](#) 后参与评论

## 深入理解 Spring 事务原理



Sacred\_Relic 2016年07月06日 11:33 1571

一、事务的基本原理 Spring事务的本质其实就是数据库对事务的支持，没有数据库的事务支持，spring是无法提供事务功能的。对于纯JDBC操作数据库，想要用到事务，可以按照以下步骤进行： 获取连...

## 《深入理解mybatis原理》 MyBatis事务管理机制

MyBatis作为Java语言的数据库框架，对数据库的事务管理是其非常重要的一个方面。本文将讲述MyBatis的事务管理的实现机制。首先介绍MyBatis的事务Transaction的接口设计以及其不...



u010349169 2014年07月20日 22:09 32304

## 深入理解Spring系列之一：开篇



tianrui Rui 2016年10月30日 20:18 2528

Spring经过大神们的构思、编码，日积月累而来，所以，对其代码的理解也不是一朝一夕就能快速完成的。源码学习是枯燥的，需要坚持！坚持！坚持！当然也需要技巧，第一遍学习的时候，不用关注全部细节，不重要的...

## 分布式缓存技术redis学习系列（六）—— 深入理解Spring Redis的使用

关于spring redis框架的使用，网上的例子很多很多。但是在自己最近一段时间的使用中，发现这些教程都是入门教程，包括很多的使用方法，与spring redis丰富的api大相径庭，真是浪费了这么...



pfnie 2016年08月27日 10:52 2080

## 【读过的书，留下的迹】Spring技术内幕——深入解析Spring架构与设计原理

前言 最近发现有时候看完一本书，时间久了容易忘记，看书不总结思考效果大打折扣，故打算写这一系列文章，一是为了整理书中的要点，帮助自己消理解；二是勉励自己多看书思考。文章中不会把书中内容讲解的非...






<div><div>linxdcn</div><div>2017年03月29日 11:00</div><div>1224</div></div>	
<div><div><div>程序员不会英语怎么行？</div><div>北大猛男教你：不背单词和语法，一个公式学好英语</div></div></div>	<div></div>
<div><div><div>909422229_SpringBoot深入理解</div><div><div>a909422229</div><div>2017年05月17日 14:04</div><div>858</div></div></div><div>spring-boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过...</div></div>	
<div><div><div>spring ioc 容器深入理解&lt;一&gt;</div><div>IoC 概述    IOC是spring的内核，Aop、声明式事务都能功能都依赖于此功能，它涉及代码解耦，设计模式，代码优化的问题的考量。ioc的初步理解    ioc的概...</div></div></div>	<div><div><div>wangqingqi20005</div><div>2016年09月08日 23:14</div><div>643</div></div></div>
<div><div><div>深入理解Spring事务原理</div><div>一、事务的基本原理        Spring事务的本质其实就是数据库对事务的支持，没有数据库的事务支持，spring是无法提供事务功能的。对于纯JDBC操作数据库，想要用到事务，可以按照以下...</div></div></div>	
<div><div><div>《深入理解mybatis原理(十)》 Mybatis插件原理简单分析</div><div>我们目前在Mybatis中，我们知道Mybatis的Mapper是一个接口，而不是一个实体类。在Java中接口是没有办法运行的。那么它是怎么运行的呢？它是通过动态代理运行。 一、MyBati...</div></div></div>	
<div><div><div><div>pfnie</div><div>2016年08月14日 10:42</div><div>1055</div></div></div></div>	
<div><div><div>struts2原理-深入理解(转)</div><div>struts2原理-深入理解 转自：http://www.cnblogs.com/yan5lang/archive/2009/07/23/1529407.html Action的定义 传统...</div></div></div>	<div><div><div>huludownload</div><div>2013年04月25日 23:30</div><div>1801</div></div></div>
<div><div><div>程序员不会英语怎么行？</div><div>老司机教你一个数学公式秒懂天下英语</div></div></div>	<div></div>
<div><div><div>个人理解的SpringAOP事务管理</div><div><div>fjnpysh</div><div>2017年03月20日 21:12</div><div>794</div></div></div></div>	



改天得好好去看看深入解析Spring的数据。 先了解AOP的相关术语: 1.通知(Advice): 通知定义了切面是什么以及何时使用。描述了切面要完成的工作和何时需要执行这个工作。 ...

深入理解windows内部原理系列--笔记二

yalecaltech 2017年04月28日 22:34 367


CR3寄存器：用来记录页目录表的物理基地址的寄存器，简称PDBR(page directory base register) 切换CR3寄存器意味着切换地址空间 不同进程拥有不同的地址...

spring的@Transactional到底是怎么工作的

maoyeqiu 2015年12月22日 00:09 1113


在这篇文章中，我们将深入了解spring的事务管理。@Transactional实际上是怎么工作的 JPA和事务管理 注意到JPA自己并不提供任何声明的类型管理是很重要的，当在一个依赖注入容器的外部使...

深入理解Spring源代码-开头篇

jp0520 2015年05月05日 12:58 1371

开篇 最近想来研究研究下spring的源代码，在此之前都是使用其部分的功能，比如springmvc，事务管理等，并未对其内部的原理进行研究过，对于这么一个经常使用的开源框架 得好好的去...

Spring事物管理理解

WMY1230 2016年11月29日 14:28 461

数据库事务概述 数据库事务(Database Transaction)，是指作为单个逻辑工作单元执行的一系列操作，要么完整地执行，要么完全不执行。 事务处理可以确保除非事务性单元...

程序员不会英语怎么行？




老司机教你一个数学公式秒懂天下英语

spring管理mybatis事务源码理解随记

zhanlanmg 2016年09月16日 06:32 1456


mysql中session和connection的关系： mysql中一个session就是一个connection，区别在于，connection是对象池中的一个可复用对象，所以它就是一个物理连接...

深入理解内存:原理简介

lujiandong1 2015年03月23日 20:49 1887

我们知道冯.诺伊曼体系结构中是把计算机划分为输入设备,输出设备,存储器,控制器,运算器. 输入设备主要是键盘,鼠标,输出设备主要是显示器,打印机 控制器与运算器我们一般全称为CPU. 存...

(4) 数据库之 深入理解 单机事务

pandajava 2015年03月13日 15:37 674

ACID 1、原子性 一个事务(transaction)中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（ Rollback ）到事务开始前的状态，就...

## 深入理解SpringMVC-基础篇



u010820702

2016年06月11日 13:57



4041

深入理解SpringMVC-入门篇 SpringMVC是一个轻量级的MVC框架，SpringMVC由于其轻量级的实现以及与Spring框架的无缝整合等诸多优势，近年来在MVC框架中脱颖而出，受到诸多开...

## 通俗易懂SpringMVC整体框架理解



ZHOUCHAOQIANG

2015年10月30日 12:55



12949

最近又重新温习了一下前台SpringMVC框架，能够从整体上对SpringMVC有一个全局的认识。在这里也总结一下，为那些即将学习SpringMVC的亲们，做一个很好的开端吧！ 1. Sprin...