

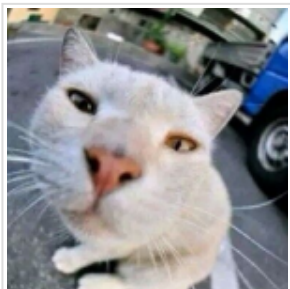


# 蓝精灵——默默争上游

改变世界的是这么一群人，他们寻找梦想中的乐园，当发现找不到是，就亲手创造了她...(专注IT的技术文章+生活随笔)

[目录视图](#)[摘要视图](#)[RSS](#) [订阅](#)

## 个人资料



蓝精灵lx



访问： 461449次

积分： 10150

等级： **BLOG > ?**

排名： 第1302名

[CSDN 2016博客之星评选结果公布](#)[【系列直播】零基础学习微信小程序!](#)[“我的2016”主题征文活动](#)[博客的神秘功能](#)

## Spring 管理事务(传播特性、隔离级别、readonly)

标签： [spring](#) [事务](#) [管理](#)

2016-07-03 23:50

508人阅读

[评论\(0\)](#)

[收藏](#)

[举报](#)

分类： [java \(466\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

### 一、事务传播机制（Propagation）

1、**spring**使用动态代理来为某个方法自动添加事务，而代理应该给哪个方法增加事务行为，是有传播机制决定的，那么它有哪些属性：

#### 1) REQUIRED

默认的，加入当前正要执行的事务不在另外一个事务里，那么就起一个新的事务。

原创： 483篇 转载： 690篇  
译文： 3篇 评论： 35条

#### 技术交流

欢迎访问我的GitHub：  
<https://github.com/lanjingling>  
联系方式：新浪微博  
<http://weibo.com/1925780475>

#### 文章分类

[java](#) (467)  
[数据库-oracle](#) (52)  
[数据库-mysql](#) (134)  
[web前端](#) (200)  
[生活](#) (15)  
[linux](#) (146)  
[nosql](#) (12)  
[安卓](#) (24)  
[算法](#) (3)  
[hadoop](#) (38)  
[mapreduce](#) (24)  
[hdfs](#) (17)  
[运维](#) (14)  
[git工具](#) (9)  
[vim](#) (1)  
[nodejs](#) (29)  
[html5](#) (18)  
[办公自动化](#) (2)  
[nginx](#) (10)  
[认证\(oauth\)](#) (4)  
[协议](#) (6)  
[python](#) (14)

比如说，ServiceB.methodB的事务级别定义为PROPAGATION\_REQUIRED,那么由于执行ServiceA.methodA的时候，ServiceA.methodA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。而假 ServiceA.methodA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。即使ServiceB.methodB的事务已经被提交，但是ServiceA.methodA在接下来fail要回滚，ServiceB.methodB也要回滚。

## 2) SUPPORTS

如果当前在事务中，即以事务的形式运行，如果当前不再一个事务中，那么就以非事务的形式运行。

## 3) MANDATORY

必须在一个事务中运行，如果没有就抛出异常。也就是说，他只能被一个父事务调用。否则，他就要抛出异常。

## 4) REQUIRES\_NEW

这个就比较绕口了。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，他才继续执行。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。

## 5) NOT\_SUPPORTED

当前不支持事务，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。比如ServiceA.methodA的PROPAGATION\_REQUIRED，而ServiceB.methodB的事务级别是PROPAGATION\_NOT\_SUPPORTED，那么当ServiceB.methodB时，ServiceA.methodA的事务挂起，而他以非事务的状态运行完，再继续ServiceA.methodA的事务。

## 6) NEVER

resin (4)  
 zookeeper (3)  
 分布式 (1)  
 mq (6)  
 jmeter (8)  
 thrift (6)  
 hbase (8)  
 hive (8)

## 文章存档

2016年12月 (24)  
 2016年11月 (33)  
 2016年10月 (27)  
 2016年09月 (17)  
 2016年08月 (33)

展开

## 阅读排行

echarts中横坐标值显示 (9143)  
 tomcat7中cookie写入中 (4919)  
 JAVA中Long与Integer比 (3857)  
 jqgrid——treegrid动态异 (3770)  
 spring使用aop时需要设置 (3206)  
 Mybatis 大数据量的批量i (2713)  
 ldap学习 (2689)  
 mybatis 对于tinyint 类型 (2651)  
 virtualbox虚拟机设置双网 (2499)  
 Communications link fail (2485)

不能在事务中运行，如果当前存在事务，则抛出异常。假设ServiceA.methodA的事务级别是PROPAGATION\_REQUIRED，而ServiceB.methodB的事务级别是PROPAGATION\_NEVER，那么ServiceA.methodA中调用ServiceB.methodB时就要抛出异常了。

## 7) NESTED

理解Nested的关键是savepoint。他与PROPAGATION\_REQUIRES\_NEW的区别是，PROPAGATION\_REQUIRES\_NEW另起一个事务，将会与他的父事务相互独立，而Nested的事务和他的父事务是相依的，他的提交是要等和他的父事务一块提交的。也就是说，如果父事务最后回滚，他也要回滚的。

而Nested事务的好处是他有一个savepoint。也就是说ServiceB.methodB失败回滚，那么ServiceA.methodA也会回滚到savepoint点上，ServiceA.methodA可以选择另外一个分支，比如ServiceC.methodC，继续执行，来尝试完成自己的事务。但是这个事务并没有在EJB标准中定义。

## 2、配置：

```
[html]
01. <!-- 事务-->
02. <beanidbeanid="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransact
03. <propertynamepropertyname="dataSource" ref="dataSource" />
04. </bean>
05. <tx:adviceidtx:adviceid="txAdvice" transaction-manager="transactionManager">
06. <tx:attributes>
07. <tx:methodnametx:methodname="add*" propagation="REQUIRED" read-only="false"/>
08. <tx:methodnametx:methodname="save*" propagation="REQUIRED" read-only="false" />
09. <tx:methodnametx:methodname="update*" propagation="REQUIRED" read-only="false" />
10. <tx:methodnametx:methodname="delete*" propagation="REQUIRED" read-only="false"/>
11. <tx:methodnametx:methodname="*" propagation="REQUIRED" /><!--read-only="true" -->
12. </tx:attributes>
13. </tx:advice>
14.
15. <aop:config>
16. <aop:pointcutidaop:pointcutid="transactionPointCut" expression="execution(*cn.edu.nuc.service..
17. <aop:advisoradvice-refaop:advisoradvice-ref="txAdvice" pointcut-ref="transactionPointCut"/>
```

- 评论排行
- wsdl和wadi区别

(3)
- mybatis 对于tinyint 类型f

(3)
- js加密原理

(3)
- ZooKeeper web管理安装

(3)
- ldap学习

(2)
- echarts中横坐标值显示不

(2)
- hadoop 文件划分, map

(2)
- XML的特殊字符处理

(2)
- easyui textbox 赋值

(2)
- Linux shell tee 命令

(1)

- 推荐文章
- \* Android 反编译初探 应用是如何被注入广告的
- \* 凭兴趣求职80%会失败, 为什么
- \* 安卓微信自动抢红包插件优化和实现
- \* 【游戏设计模式】之四 《游戏编程模式》全书内容提炼总结
- \* 带你开发一款给Apk中自动注入代码工具icodetools(完善篇)

- 最新评论
- wsdl和wadi区别

yuruixin\_china: good
- echarts中横坐标值显示不全(自动

蓝精灵lx: @zwxscience:赞
- echarts中横坐标值显示不全(自动

```
18. </aop:config>
```

二、事务隔离级别（Isolation）

1、这里的事务隔离级别和数据库的隔离级别是一个意思。数据库的隔离级别（MySQL）有4个：

读数据一致性及允许的并发副作 用 隔离级别	读数据一致性	脏读	不可重 复读	幻读
未提交读（Read uncommitted）	最低级别，只能保证不读取物理上损坏的数据，事务可以看到其他事务没有被提交的数据（脏数据）	是	是	是
已提交读（Read committed）	语句级，事务可以看到其他事务已经提交的数据	否	是	是
可重复读（Repeatable read）	事务级，事务中两次查询的结果相同	否	否	是
可序列化（Serializable）	最高级别，事务级。顺序执行	否	否	否

spring控制事务隔离级别，属性有：

1) DEFAULT （默认）

这是一个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。

2) READ\_UNCOMMITTED （读未提交）

这是事务最低的隔离级别，它允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。

3) READ\_COMMITTED （读已提交）

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。这种事务隔离级别可以避免脏读出现，但是可能会出现不可重复读和幻像读。

**zwxscience:** series 添加  
showAllSymbol:true, 也可以

**wsdl和wadi区别**  
**hr1986sd:** 好文章啊!

**JedisPool高并发**  
**hzyImf:** 博主, 新版的Jedis3.0没有returnResource方法了, 看github里面的test c...

**ZooKeeper web管理安装node-zl**  
**yosaku01:** 貌似zookeeper那个包不用装也可以启动

**hadoop 文件划分, map执行浅析**  
**qiuzhideyimaoqian:** 你好, 有个问题请教下, 《Split的Start、Length都好说, 都是划分前就定好的。而Split...

**hadoop 文件划分, map执行浅析**  
**qiuzhideyimaoqian:** 你好, 有个问题请教下, 《Split的Start、Length都好说, 都是划分前就定好的。而Split...

**ZooKeeper web管理安装node-zl**  
**蓝精灵lx:** @u011126891:不用修改, 只需要该start.sh中zk的地址即可。如果想加权限, 需要把用户...

**ZooKeeper web管理安装node-zl**  
**yosaku01:** package.json你没有修改吧?



#### 4) REPEATABLE\_READ (可重复读)

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。它除了保证一个事务不能读取另一个事务未提交的数据外, 还保证了不可重复读

#### 5) SERIALIZABLE (串行化)

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。除了防止脏读, 不可重复读外, 还避免了幻像读。

### 2、配置:

```
[html]
01. <!-- 事务-->
02. <beanidbeanid="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransact.
03. <propertynamepropertyname="dataSource" ref="dataSource" />
04. </bean>
05. <tx:adviceidtx:adviceid="txAdvice" transaction-manager="transactionManager">
06. <tx:attributes>
07. <tx:methodnametx:methodname="add*" propagation="REQUIRED" isolation="DEFAULT"/>
08. <tx:methodnametx:methodname="save*" propagation="REQUIRED" isolation="DEFAULT"/>
09. <tx:methodnametx:methodname="update*" propagation="REQUIRED" isolation="DEFAULT" />
10. <tx:method name="delete*" propagation="REQUIRED" isolation="DEFAULT"/>
11. <tx:methodnametx:methodname="*" propagation="REQUIRED" /><!--read-only="true" -->
12. </tx:attributes>
13. </tx:advice>
14.
15. <aop:config>
16. <aop:pointcutidaop:pointcutid="transactionPointCut" expression="execution(*cn.edu.nu
17. <aop:advisoradvice-refaop:advisoradvice-ref="txAdvice" pointcut-ref="transactionPointCut"/>
18. </aop:config>
```

### 三、只读事务 (readonly)

“只读事务”并不是一个强制选项，它只是一个“暗示”，提示数据库驱动程序和数据库系统，这个事务并不包含更改数据的操作，那么JDBC驱动程序和数据库就有可能根据这种情况对该事务进行一些特定的优化，比方说不安排相应的数据库锁，以减轻事务对数据库的压力，毕竟事务也是要消耗数据库的资源的。

```
[html] C 8

01. <!-- 事务-->
02. <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransact
03. <property name="dataSource" ref="dataSource" />
04. </bean>
05. <tx:advice id="txAdvice" transaction-manager="transactionManager">
06. <tx:attributes>
07. <tx:method name="add*" propagation="REQUIRED" read-only="false" rollback-
    for="java.lang.Exception" />
08. <tx:method name="save*" propagation="REQUIRED" read-only="false" rollback-
    for="java.lang.Exception" />
09. <tx:method name="update*" propagation="REQUIRED" read-only="false" rollback-
    for="java.lang.Exception" />
10. <tx:method name="delete*" propagation="REQUIRED" read-only="false" rollback-
    for="java.lang.Exception" />
11. <tx:method name="*" propagation="REQUIRED" /><!-- read-only="true" -->
12. </tx:attributes>
13. </tx:advice>
14.
15. <aop:config>
16. <aop:pointcut id="transactionPointCut" expression="execution(*cn.edu.nu
    (...))" />
17. <aop:advisor advice-ref="txAdvice" pointcut-ref="transactionPoin
18. </aop:config>
```

顶 踩  
0 0

上一篇 [linux下压缩与解压\(zip、unzip、tar\)详解](#)

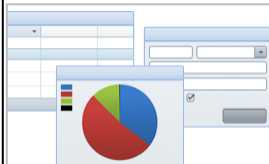
下一篇 [ExecutorService.invokeAny\(\)和ExecutorService.invokeAll\(\)的使用剖析](#)

## 我的同类文章

### java (466)

- [java实现定时任务的三种方法](#) 2016-12-28 阅读 49
- [浅谈Java中CyclicBarrier的用法](#) 2016-12-28 阅读 42
- [FileChannel](#) 2016-12-26 阅读 36
- [java版“本佛祖保佑永无bug”](#) 2016-12-21 阅读 103
- [通过Spring @PostConstruct ...](#) 2016-12-19 阅读 335
- [在web项目启动时，执行某个...](#) 2016-12-15 阅读 79
- [spring task 定时任务实现](#) 2016-12-28 阅读 43
- [如何重置一个ArrayList--clear ...](#) 2016-12-28 阅读 55
- [Java高效读取大文件](#) 2016-12-26 阅读 38
- [图解JDK7的Comparison met...](#) 2016-12-20 阅读 28
- [java工厂模式](#) 2016-12-17 阅读 78

[更多文章](#)



jQuery MiniUI



快速 开发  
Java、.Net 软件

[参考知识库](#)

**MySQL知识库**

18105 关注 | 1442 收录

**Java SE知识库**

19976 关注 | 468 收录

**Java EE知识库**

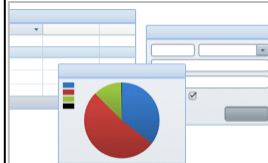
13487 关注 | 1215 收录

**Java 知识库**

21719 关注 | 1436 收录

## 猜你在找

springMvc jdbc jQWidgets项目案例jasperreport自动化报表系    Hibernate中文参考文档JFIS  
Redis内存管理和优化    Hibernate参考文档 304  
spring3.2入门到大神（备java基础、jsp、servlet，javaee精髓    Spring事务的传播特性和隔离级别  
精通memcached数据库管理深度讲解    spring事务隔离级别及传播特性  
精通mysql服务器端编程    Spring事务的传播特性和隔离级别



# jQuery MiniUI



快速开发WEB软  
Java、.Net

[查看评论](#)

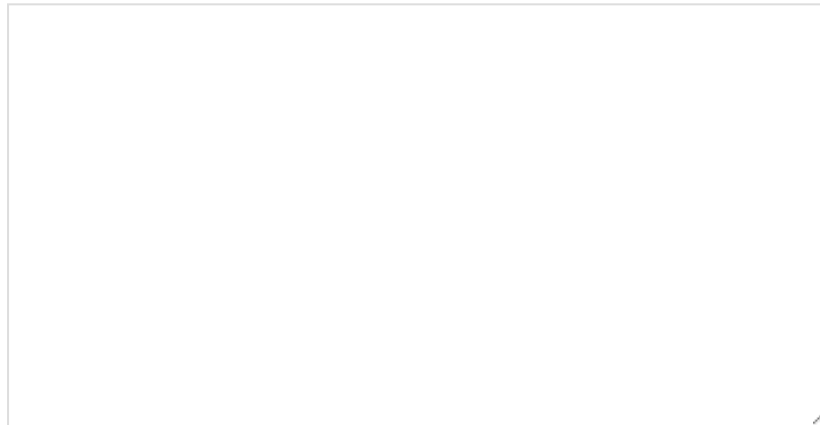
暂无评论

[发表评论](#)

用户 名: andilyliao

评论内容:





提交

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

#### 核心技术类目

全部主题   Hadoop   AWS   移动游戏   Java   Android   iOS   Swift   智能硬件   Docker   OpenStack   VPN  
Spark   ERP   IE10   Eclipse   CRM   JavaScript   数据库   Ubuntu   NFC   WAP   jQuery   BI   HTML5  
Spring   Apache   .NET   API   HTML   SDK   IIS   Fedora   XML   LBS   Unity   Splashtop   UML  
components   Windows Mobile   Rails   QEMU   KDE   Cassandra   CloudStack   FTC   coremail   OPhone  
CouchBase   云计算   iOS6   Rackspace   Web App   SpringSide   Maemo   Compuware   大数据   aptech  
Perl   Tornado   Ruby   Hibernate   ThinkPHP   HBase   Pure   Solr   Angular   Cloud Foundry   Redis  
Scala   Django   Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知

司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

