



## 《Spring Boot 基础教程》

## 第 15 节

## 使用 SQL 数据库-事务处理

## 一、事务有四个特性：ACID

原子性 (Atomicity)：事务是一个原子操作，由一系列动作组成。事务的原子性确保动作要么全部完成，要么完全不起作用。

一致性 (Consistency)：一旦事务完成（不管成功还是失败），系统必须确保它所建模的业务处于一致的状态，而不会是部分完成部分失败。在现实中的数据不应该被破坏。

隔离性 (Isolation)：可能有许多事务会同时处理相同的数据，因此每个事务都应该与其他事务隔离开来，防止数据损坏。

持久性 (Durability)：一旦事务完成，无论发生什么系统错误，它的结果都不应该受到影响，这样就能从任何系统崩溃中恢复过来。通常情况下，事务的结果被写到持久化存储器中。

## 二、传播行为

当事务方法被另一个事务方法调用时，必须指定事务应该如何传播。例如：方法可能继续在现有事务中运行，也可能开启一个新事务，并在自己的事务中运行。

Spring 定义了七种传播行为：

PROPAGATION\_REQUIRED 表示当前方法必须运行在事务中。如果当前事务存在，方法将会在该事务中运行。否则，会启动一个新的事务，**Spring 默认使用**

PROPAGATION\_SUPPORTS 表示当前方法不需要事务上下文，但是如果存在当前事务的话，那么该方法会在这个事务中运行

PROPAGATION\_MANDATORY 表示该方法必须在事务中运行，如果当前事务不存在，则会抛出一个异常

PROPAGATION\_REQUIRED\_NEW 表示当前方法必须运行在它自己的事务中。一个新的事务将被启动。如果存在当前事务，在该方法执行期间，当前事务会被挂起。如果使用 JTATransactionManager 的话，则需要访问 TransactionManager

PROPAGATION\_NOT\_SUPPORTED 表示该方法不应该运行在事务中。如果存在当前事务，在该方法运行期间，当前事务将被挂起。如果使用 JTATransactionManager 的话，则需要访问 TransactionManager

PROPAGATION\_NEVER 表示当前方法不应该运行在事务上下文中。如果当前正有一个事务在运行，则会抛出异常

PROPAGATION\_NESTED 表示如果当前已经存在一个事务，那么该方法将会在嵌套事务中运行。嵌套的事务可以独立于当前事务进行单独地提交或回滚。如果当前事务不存在，那么其行为与

PROPAGATION\_REQUIRED 一样。注意各厂商对这种传播行为的支持是有所差异的。可以参考资源管理器的文档来确认它们是否支持嵌套事务

## 三、隔离级别

隔离级别定义了一个事务可能受其他并发事务影响的程度。

ISOLATION\_DEFAULT 使用后端数据库默认的隔离级别，**Spring 默认使用**，mysql 默认的隔离级别为：

**Repeatable Read(可重复读)**

ISOLATION\_READ\_UNCOMMITTED 读未提交，最低的隔离级别，允许读取尚未提交的数据变更，可能会导致脏读、幻读或不可重复读

ISOLATION\_READ\_COMMITTED 读已提交，允许读取并发事务已经提交的数据，可以阻止脏读，但是幻读



或不可重复读仍有可能发生

ISOLATION\_REPEATABLE\_READ 可重复读，对同一字段的多次读取结果都是一致的，除非数据是被本身事务自己所修改，可以阻止脏读和不可重复读，但幻读仍有可能发生

ISOLATION\_SERIALIZABLE 可串行化，最高的隔离级别，完全服从 ACID 的隔离级别，确保阻止脏读、不可重复读以及幻读，也是最慢的事务隔离级别，因为它通常是通过完全锁定事务相关的数据库表来实现的

脏读（Dirty reads）——脏读发生在一个事务读取了另一个事务改写但尚未提交的数据时。如果改写再稍后被回滚了，那么第一个事务获取的数据就是无效的。

不可重复读（Nonrepeatable read）——不可重复读发生在一个事务执行相同的查询两次或两次以上，但是每次都得到不同的数据时。这通常是因为另一个并发事务在两次查询期间进行了更新。

幻读（Phantom read）——幻读与不可重复读类似。它发生在一个事务（T1）读取了几行数据，接着另一个并发事务（T2）插入了一些数据时。在随后的查询中，第一个事务（T1）就会发现多了一些原本不存在的记录。

#### 四、属性说明 @Transactional

a、isolation：用于指定事务的隔离级别。默认为底层事务的隔离级别。

b、noRollbackFor：指定遇到指定异常时强制不回滚事务。

c、noRollbackForClassName：指定遇到指定多个异常时强制不回滚事务。该属性可以指定多个异常类名。

d、propagation：指定事务的传播属性。

e、readOnly：指定事务是否只读。表示这个事务只读取数据但不更新数据，这样可以帮助数据库引擎优化事务。若真的是一个只读取的数据库应设置 readOnly=true

f、rollbackFor：指定遇到指定异常时强制回滚事务。

g、rollbackForClassName：指定遇到指定多个异常时强制回滚事务。该属性可以指定多个异常类名。

h、timeout：指定事务的超时时长。

#### 注意：

1.mysql 为例，存储引擎不能使用 MyISAM，应该使用 InnoDB

#### 实现代码

```
@Service
```

```
public class UserService {
```

```
    @Autowired
```

```
    private RoncooUserDao roncooUserDao;
```

```
    @Autowired
```

```
    private RoncooUserLogDao roncooUserLogDao;
```

```
    /**
```

```
     * 用户注册
```

```
     *
```

```
     * @return
```



```
*/
@Transactional
public String register(String name, String ip) {
    // 1.添加用户
    RoncooUser roncooUser = new RoncooUser();
    roncooUser.setName(name);
    roncooUser.setCreateTime(new Date());
    roncooUserDao.insert(roncooUser);

    // 测试使用
    boolean flag = true;
    if (flag) {
        throw new RuntimeException();
    }

    // 2.添加注册日志
    RoncooUserLog roncooUserLog = new RoncooUserLog();
    roncooUserLog.setUserName(name);
    roncooUserLog.setUserIp(ip);
    roncooUserLog.setCreateTime(new Date());
    roncooUserLogDao.save(roncooUserLog);

    return "success";
}
}
```

## 测试

```
@Autowired
private UserService userService;

@Test
public void register() {
    String result = userService.register("无境", "192.168.1.1");
    System.out.println(result);
}
```

更多课程信息，请关注 龙果学院 官方网站 <http://www.roncoo.com/>

或关注 龙果 微信公众号 RonCoo\_com

