## Transaction事物

1. 何为Spring事物
2. 事物满足的4个特性：ACID

* 原子性 A：组成事物的模块不能在分割
* 一致性 C：要么都成功，要么都失败
* 隔离性 I：5大隔离级别，默认、读未提交、读已提交、可重复读、序列化读
* 持久性 D：保存到数据库中

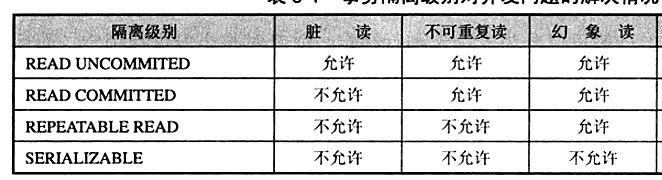
1. 数据并发问题

* 脏读：读到了未提交的数据
* 不可重复读：每次读的数据不一致，发生更新操作
* 幻（想）读：读到数据条数，多了或少了。发生新增、或是删除。

1. 数据库锁机制

数据库通过锁的机制，解决在并发访问的问题，一般分为**表锁定**和**行锁定**（乐观锁和悲观锁）。

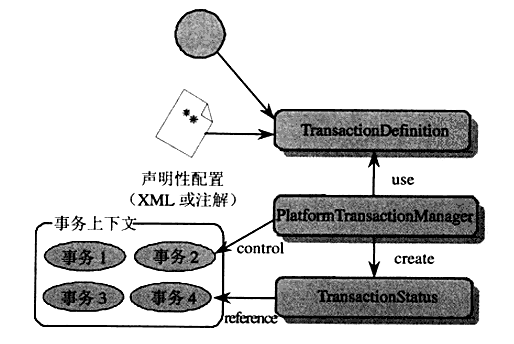
1. 事物的隔离级别



1. ThreadLocal基础知识
2. 在多线程情况下，访问对象必须增加线程同步，但模版类并没有类似的操作，因为线程同步会降低并发性。影响系统的性能。Spring的模版类，使用ThreadLocal，解决线程安全的问题。
3. ThreadLocal不是线程，他是线程的一个本地化对象。当多个线程使用同一个对象的时候，ThreadLocal会为每一个线程创建一个对象副本。所以从线程的角度来看，这个对象（变量）就是线程的本地变量。
4. ThreadLocal类内部有一个Map，用于保存每一个线程的变量副本，Map中的Key=线程对象；对应的Value=对象（变量）副本。

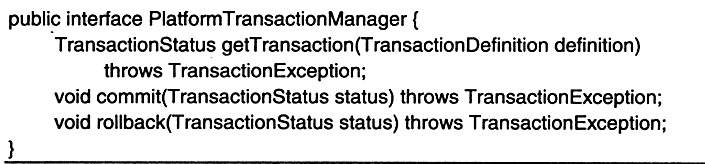
Eg：com.springtx.threadlocal.ThradLocalTests

1. Spring对事物管理的支持
2. Spring事物管理使用TransactionTemplate。
3. Spring的事物管理亮点在于**声明式事物管理**。
4. 关于事物管理的抽象：



（Spring事物管理：Service Provider Interface）

1. TransactionDefinition：定义事物的属性：隔离级别，超时时间，事务的传播规则等。
2. TransactionDefinition：代表事物的运行状态
3. PlatformTransactionManager：



1. Spring事物管理器的实现：



1. 事物传播行为

当我们调用一个基于Spring的Service接口方法时，将运行在Spring的事物管理中，Service中的方法，完全有可能去调用其他Service的其他method。因此会产生随着方法调用而引发的事物传播行为。

TransactionDefinition定义了事物传播行为。



1. 编程式事物
2. 实际应用中，很少使用编程来进行事物的管理。但是Spring还是提供了相关模版，用于特殊场景实现编程式事物。
3. 步骤：

|  |
| --- |
| 1、数据源、JdbcTemplate、TransactionTemplate配置  <!-- 基于占位符的方式，通过属性文件，加载数据库连接资源 -->  <context:property-placeholder location=*"classpath:jdbc.properties"* />  <!-- 配置数据源 -->  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*  destroy-method=*"close"* p:driverClassName=*"${jdbc.driverClassName}"*  p:url=*"${jdbc.url}"* p:username=*"${jdbc.username}"* p:password=*"${jdbc.password}"* />  <!-- 配置基于JDBC的模版 -->  <bean id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*  p:dataSource-ref=*"dataSource"* />  <!-- 配置事物管理器-->  <bean id=*"txManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*  p:dataSource-ref=*"dataSource"* />  <!-- 配置事物管理模版 -->  <bean id=*"transactionTemplate"*  class=*"org.springframework.transaction.support.TransactionTemplate"*  p:transactionManager-ref=*"txManager"*/> |
| 2、配置Dao和Service  <!-- Dao中注入jdbcTemplate -->  <bean id=*"forumDao"* class=*"com.springtx.dao.jdbc.ForumJdbcDao"*  p:jdbcTemplate-ref=*"jdbcTemplate"* />  <!-- Service中注入transactionTemplate -->  <bean id=*"forumService"* class=*"com.springtx.service.ForumService"*  p:forumDao-ref=*"forumDao"* p:template-ref=*"transactionTemplate"* /> |
| **3、编程式事物**  **public** **class** ForumService {  **private** ForumDao forumDao;  **private** TransactionTemplate template;  **public** **void** setForumDao(ForumDao forumDao) {  **this**.forumDao = forumDao;  }  **public** **void** setTemplate(TransactionTemplate template) {  **this**.template = template;  }  **public** **void** addForum(**final** Forum forum) {  template.execute(**new** TransactionCallbackWithoutResult() {  **protected** **void** doInTransactionWithoutResult(TransactionStatus status) {  forumDao.addForum(forum);  }  });  }  } |
| 1. **Dao层接口和实现**   **public** **interface** ForumDao {  **void** addForum(Forum forum);  }  **public** **class** ForumJdbcDao **extends** JdbcDaoSupport **implements** ForumDao {  **public** **void** addForum(**final** Forum forum) {  **final** String sql = "INSERT INTO t\_forum(forum\_id,forum\_name,forum\_desc) VALUES(?,?,?)";    getJdbcTemplate().update(**new** PreparedStatementCreator() {    **public** PreparedStatement createPreparedStatement(Connection conn)  **throws** SQLException {  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, forum.getForumId());  ps.setString(2, forum.getForumName());  ps.setString(3, forum.getForumDesc());  **return** ps;  }  });  }  } |
| 5、测试类  @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations = { "/applicationContext.xml" })  **public** **class** ForumServiceTests {  @Autowired  **private** ApplicationContext context;  @Test  **public** **void** testAddForum() {  //ForumService forumService = context.getBean(ForumService.class);  ForumService forumService = (ForumService) context.getBean("forumService");  Forum forum = **new** Forum();  forum.setForumId(8);  forum.setForumDesc("desc");  forum.setForumName("name");  forumService.addForum(forum);  }  } |

1. 使用XML配置声明式事物
2. 大多数用户选择声明式事物管理，对代码的侵入性最小，可以让事物管理代码完全从业务中剥离。符合非侵入性轻量级的概念。
3. Spring声明式事物通过Spring AOP实现，Spring负责将事物管理增强逻辑动态织入到业务方法相应的连接点上。
4. 步骤：

|  |
| --- |
| 1、定义配置  <!-- 目标对象 -->  <bean id=*"forumService2Target"* class=*"com.springtx.service.ForumService2"*  p:forumDao-ref=*"forumDao"* />  <!-- 织入了增强（声明式事物）的代理对象 -->  <bean id=*"forumService2"*  class=*"org.springframework.transaction.interceptor.TransactionProxyFactoryBean"*  p:transactionManager-ref=*"txManager"* p:target-ref=*"forumService2Target"*>  <!-- 事物属性配置 -->  <property name=*"transactionAttributes"*>  <props>  <!-- addForum方法：在抛出PessimisticLockingFailureException异常，依然提交事物 -->  <prop key=*"addForum"*>  PROPAGATION\_REQUIRED,+PessimisticLockingFailureException  </prop>  <!-- get开头方法：只读事物 -->  <prop key=*"get\*"*>PROPAGATION\_REQUIRED,readOnly</prop>  <!-- 所有方法： -->  <prop key=*"\*"*>PROPAGATION\_REQUIRED,-tion</prop>  </props>  </property>  </bean> |
| **2、业务类，纯POJO**  **public** **class** ForumService2 {  **private** ForumDao forumDao;    **public** **void** setForumDao(ForumDao forumDao) {  **this**.forumDao = forumDao;  }  **public** ForumDao getForumDao() {  **return** forumDao;  }  **public** **void** addForum(Forum forum) {    forumDao.addForum(forum);    /\*\*  \* 模拟异常：根据配置，事物依然会提交  \*/  /\*if (true) {  throw new PessimisticLockingFailureException("addForum-PessimisticLockingFailureException!!!");  }\*/    /\*\*  \* 模拟异常：根据配置，事物会回滚  \*/  /\*if (true) {  throw new NullPointerException("addForum-NullPointerException");  }\*/    }  } |
| 3、测试类  /\*\*  \* 基于配置的-声明式事物测试  \*/  @Test  **public** **void** testAddForum() {  ForumService2 forumService2 = (ForumService2) context.getBean("forumService2");  Forum forum = **new** Forum();  forum.setForumId(19);  forum.setForumDesc("desc");  forum.setForumName("name");  forumService2.addForum(forum);  } |

1. 使用注解配置声明式事物

基于tx/aop命名空间的配置

|  |
| --- |
| 1、配置  <context:property-placeholder location=*"classpath:jdbc.properties"* />  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*  destroy-method=*"close"* p:driverClassName=*"${jdbc.driverClassName}"*  p:url=*"${jdbc.url}"* p:username=*"${jdbc.username}"* p:password=*"${jdbc.password}"* />  <bean id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*  p:dataSource-ref=*"dataSource"* />  <bean id=*"txManager"*  class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*  p:dataSource-ref=*"dataSource"* />  <!--对标注@Transactional注解的bean，进行加工处理，比边组织事物管理切面 -->  <!-- proxy-target-class=true,说明使用CGLib动态代理 -->  <tx:annotation-driven transaction-manager=*"txManager"*  proxy-target-class=*"true"* />  <bean id=*"forumDao"* class=*"com.springtx.dao.ForumDaoImpl"*  p:jdbcTemplate-ref=*"jdbcTemplate"* />    <!-- forumService3标注了@Transactional，所以会被注解驱动，自动织入事物！-->  <bean id=*"forumService3"* class=*"com.springtx.service.ForumService3"*  p:forumDao-ref=*"forumDao"* /> |
| 2、业务实现类  @Transactional(propagation = Propagation.*REQUIRED*, isolation = Isolation.*READ\_COMMITTED*, noRollbackFor = { PessimisticLockingFailureException.**class** }, rollbackFor = { NullPointerException.**class** })  **public** **class** ForumService3 {  **private** ForumDao forumDao;  **public** **void** setForumDao(ForumDao forumDao) {  **this**.forumDao = forumDao;  }  **public** ForumDao getForumDao() {  **return** forumDao;  }  **public** **void** addForum(Forum forum) {  forumDao.addForum(forum);  /\*\*  \* 模拟异常：根据配置，事物依然会提交  \*/  /\*if (true) {  throw new PessimisticLockingFailureException(  "addForum-PessimisticLockingFailureException!!!");  }\*/  /\*\*  \* 模拟异常：根据配置，事物会回滚  \*/  /\*if (true) {  throw new NullPointerException("addForum-NullPointerException");  }\*/  }  } |
| 3、测试类  /\*\*  \* 基于注解的-声明式事物测试  \*/  @Test  **public** **void** testAddForum() {  ForumService3 forumService =context.getBean("forumService3",ForumService3.**class**);  Forum forum = **new** Forum();  forum.setForumId(20);  forum.setForumDesc("desc");  forum.setForumName("name");  forumService.addForum(forum);  } |

1. 总结

Spring声明式事物是Spring中的亮点，以前在厚重的EJB才能享受的功能，现在Spring轻量级框架中即可享用。

Spring事物管理，讲业务代码从事物代码中剥离出来，代码复杂度大大降低。被织入的事物代码基于Spring事物同步管理器进行工作。事物同步管理器维护这业务类对象线程相关的资源。事物管理器是线程安全的，因为他是基于ThreadLocal实现。

Spring事物管理相对简单，关注二点：1、切点信息，用于定位业务类实施事物切面的业务方法；2、控制事物行为的事物属性，包括隔离级别、传播行为、是否只读、超时时间、回滚机制等。

Spring3.0基于tx/aop schema命名空间，实现注解声明式事物，大大简化了配置的难度。