# MySQL事务

## 事务的概念

事务指逻辑上的一组操作，组成这组操作的各个单元，要么全部成功，要么全部不成功。

例如：A——B转帐，对应于如下两条sql语句

update account set money=money-100 where name=‘a’;

update account set money=money+100 where name=‘b’;

## 数据库开启事务命令DTL

* mysql引擎是支持事务的
* mysql默认自动提交事务。每条语句都处在单独的事务中。
* 事务操作
  + **start transaction 开启事务**
  + **Rollback 回滚事务**
  + **Commit 提交事务**

## JDBC使用事务

当Jdbc程序向数据库获得一个Connection对象时，默认情况下这个Connection对象会自动向数据库提交在它上面发送的SQL语句。若想关闭这种默认提交方式，让多条SQL在一个事务中执行，可使用下列语句：

**JDBC控制事务语句**

Connection.setAutoCommit(false);start transaction

Connection.rollback(); rollback

Connection.commit(); commit

|  |
| --- |
|  |

## 事务的特性（面试题）

**原子性：**指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

**一致性：**事务必须使数据库从一个一致性状态变换到另外一个一致性状态。转账前和转账后的总金额不变。

**隔离性：**事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。

**持久性：**指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响。

## 事务的隔离级别

**赃读：**指一个事务读取了另一个事务未提交的数据。

**不可重复读：**在一个事务内读取表中的某一行数据，多次读取结果不同。一个事务读取到了另一个事务提交后的数据。（update）

**虚读（幻读）：**是指在一个事务内读取到了别的事务插入的数据，导致前后读取不一致。 (insert)

**数据库通过设置事务的隔离级别防止以上情况的发生：**

\* 1、READ UNCOMMITTED: 赃读、不可重复读、虚读都有可能发生。

\* 2、READ COMMITTED: 避免赃读。不可重复读、虚读都有可能发生。（oracle默认的）

\* 4、REPEATABLE READ:避免赃读、不可重复读。虚读有可能发生。（mysql默认）

\* 8、SERIALIZABLE: 避免赃读、不可重复读、虚读。

**级别越高，性能越低，数据越安全**

mysql中：

查看当前的事务隔离级别：SELECT @@TX\_ISOLATION;

更改当前的事务隔离级别：SET TRANSACTION ISOLATION LEVEL 四个级别之一。

设置隔离级别必须在事务之前

## 隔离级别演示

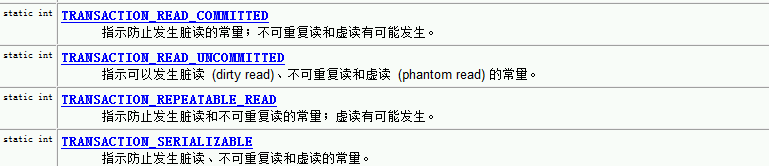
练习：设置事务的隔离级别为 read uncommitted

* set transaction isolation level read uncommitted;

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 线程1 | 线程2 | 说明 |
| t1 | begin; |  |  |
| t2 | select \* from account where name='zs';  结果1000块 |  |  |
| t3 |  | begin; |  |
| t4 |  | update account set money=money+100 where name='zs'; |  |
| t5 | select \* from account where name='zs';  结果1100块 |  | 读到了另一个线程未提交事务的数据。赃读发生了 |
| t6 |  | commit; |  |
| t7 | select \* from account where name='zs';  结果1100块 |  | 读到了另一个线程提交事务的update数据。不可重复读发生了 |
| t8 |  | insert into account values(4,'zl',1000);  执行insert语句插入数据，事务自动提交了 |  |
| t9 | select \* from account;  查到4条数据 |  | 读到了另一个线程自动提交事务的insert语句数据。虚读发生了 |
| t10 | commit; |  |  |

## JDBC控制事务的隔离级别

Connection接口：

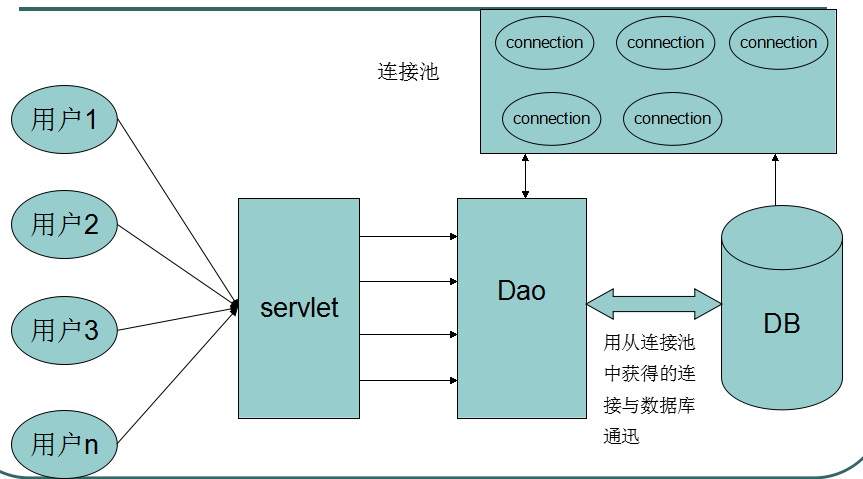


设置隔离级别：必须在开启事务之前。

Connection.setTransactionIsolation(int level);

# 数据库连接池

## 连接池原理：（面试）



目的：解决建立数据库连接耗费资源和时间很多的问题，提高性能。

## 常用的数据库连接池

DBCP（DataBase Connection Pool）数据库连接池，是java数据库连接池的一种，由Apache开发，通过数据库连接池，可以让程序**自动管理数据库连接的释放和断开**。

C3P0是一个开源的JDBC连接池，它实现了数据源和JNDI绑定，支持JDBC3规范和JDBC2的标准扩展。目前使用它的开源项目有Hibernate，Spring等。

## DBCP使用

使用步骤：

> 添加jar包 commons-dbcp-1.4.jar commons-pool-1.5.6.jar

> 添加属性资源文件

|  |
| --- |
| 在src中创建dbconfig.properties文件 |
| #连接设置  driverClassName=com.mysql.jdbc.Driver  url=jdbc:mysql://localhost:3306/day16  username=root  password=  #<!-- 初始化连接 -->  initialSize=10  #最大连接数量  maxActive=50  #<!-- 最大空闲连接 -->  maxIdle=20  #<!-- 最小空闲连接 -->  minIdle=5  #<!-- 超时等待时间以毫秒为单位 6000毫秒/1000等于60秒 -->  maxWait=60000  #JDBC驱动建立连接时附带的连接属性属性的格式必须为这样：[属性名=property;]  #注意："user" 与 "password" 两个属性会被明确地传递，因此这里不需要包含他们。  connectionProperties=useUnicode=true;characterEncoding=utf8  #指定由连接池所创建的连接的自动提交（auto-commit）状态。  defaultAutoCommit=true  #driver default 指定由连接池所创建的连接的只读（read-only）状态。  #如果没有设置该值，则“setReadOnly”方法将不被调用。（某些驱动并不支持只读模式，如：Informix）  defaultReadOnly=  #driver default 指定由连接池所创建的连接的事务级别（TransactionIsolation）。  #可用值为下列之一：（详情可见javadoc。）NONE,READ\_UNCOMMITTED, READ\_COMMITTED, REPEATABLE\_READ, SERIALIZABLE  defaultTransactionIsolation=REPEATABLE\_READ |

> 编写数据源工具类

|  |
| --- |
|  |

## C3P0的使用

使用步骤：

1、添加jar包

2、编写配置文件c3p0-config.xml，放在classpath中，或classes目录中

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <c3p0-config>  <default-config>  <property name="driverClass">com.mysql.jdbc.Driver</property>  <property name="jdbcUrl">jdbc:mysql://localhost:3306/day12</property>  <property name="user">root</property>  <property name="password">123456</property>  <property name="initialPoolSize">10</property>  <property name="maxIdleTime">30</property>  <property name="maxPoolSize">100</property>  <property name="minPoolSize">10</property>  </default-config>  </c3p0-config> |

3、编写工具类

|  |
| --- |
|  |

# JNDI

## JNDI概念

* JNDI(Java Naming and Directory Interface)是一个应用程序设计的API
* 为开发人员提供了查找和访问各种命名和目录服务的通用、统一的接口
* 类似JDBC都是构建在抽象层上。
* 现在JNDI已经成为J2EE的标准之一，所有的J2EE容器都必须提供一个JNDI的服务。

## Tomcat中配置JNDI 数据源

* 开发JavaWeb应用，必须使用一个JavaWeb服务器，JavaWeb服务器都内置数据源。这种配置就是JDNI
* **Tomcat 内置了数据源就是：（DBCP）,所以数据源只需要配置服务器即可**。

|  |
| --- |
|  |

## 配置数据源的步骤：

* 参照Tomcat的文件说明

|  |
| --- |
|  |

### Step1:拷贝数据库连接驱动的jar到tomcat的lib目录下



注意：如果出现java.lang.AbstractMethodError: com.mysql.jdbc.Connection.isValid(I)Z的问题，使用更高版本的数据库连接驱动

### Step2配置数据源XML文件

a)如果把配置信息写在tomcat下的conf目录的context.xml中，那么所有应用都能使用此数据源。

b)如果是在当前应用的META-INF中创建context.xml, 编写数据源，那么只有当前应用可以使用。

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <Context>  <Resource name=*"jdbc/day12DB"*  auth=*"Container"*  type=*"javax.sql.DataSource"*  username=*"root"*  password=*"123456"*  driverClassName=*"com.mysql.jdbc.Driver"*  url=*"jdbc:mysql://localhost:3306/day12"*  maxTotal=*"8"*  maxIdle=*"4"*/>  </Context> |

### Step3使用连接池

|  |
| --- |
|  |

### 多个数据源的配置

* 声明多个Resource即可