JDBC升级之连接池

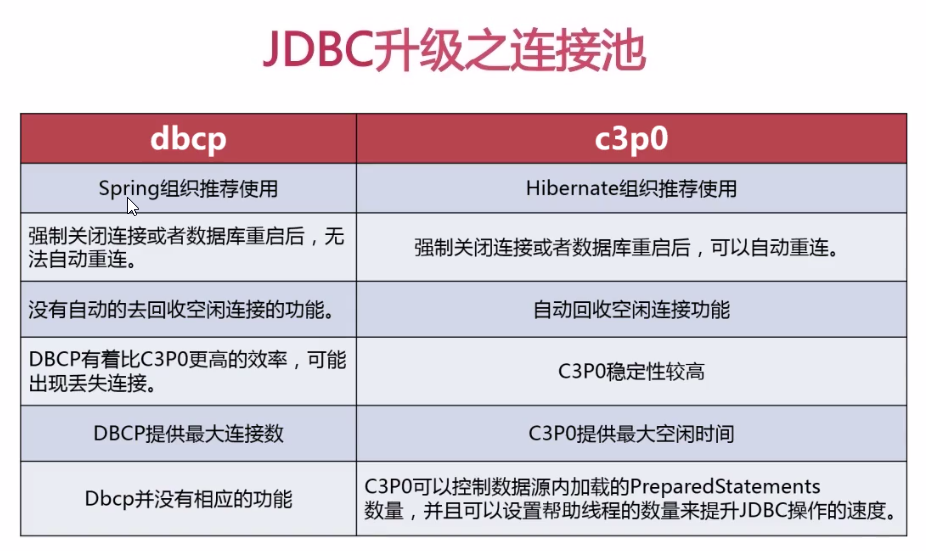
1、数据库连接是一种重要的资源。

2、频繁的连接数据库会增加数据库的压力。

常用的开源数据库连接池

dbcp

c3p0



事务管理

事务（TRANSACTION）是作为单个逻辑工作单元执行的一系列操作。

这些操作作为一个整体一起向系统提交，要么都执行，要么都不执行。

事务特点

1. 原子性，事务是一个完整的操作。
2. 一致性，当事务完成时，数据必须处于一致的状态。
3. 隔离性，对数据进行修改的所有并发事务是彼此隔离的。
4. 永久性，事务完成后，它对数据库的修改被永久保持。

JDBC对事务管理的支持

1、通过提交commit()或是回退rollback()来管理事务的操作

2、事务默认是自动提交的。

3、可以通过调用setAutoCommit(false)来禁止自动提交。

JDBC调用无参存储过程

存储过程代码：

CREATE PROCEDURE sun.sp\_select\_nofilter()

BEGIN

select \* from person;

END;

调用代码

public static void select\_nofilter() throws SQLException

{

//1、获得连接

Connection conn = DBUtil.getConnection();

//2、获得callablestatement

CallableStatement cs = conn.prepareCall("call sp\_select\_nofilter()");

//3、执行存储过程

cs.execute();

//4、处理返回的结果:结果集，出参

ResultSet res = cs.getResultSet();

while(res.next())

{

System.out.println(res.getString("user\_name")+" "+

res.getString("email")+" "+res.getString("mobile"));

}

}

JDBC带输入参数存储过程

存储过程代码

BEGIN

IF sp\_name IS NULL OR sp\_name ='' THEN

SELECT \* FROM person;

ELSE

IF LENGTH(sp\_name)=11 AND SUBSTRING(sp\_name,1,1)=1 THEN

SELECT \* FROM person WHERE mobile=sp\_name;

ELSE

SELECT \* FROM person WHERE user\_name LIKE CONCAT('%',sp\_name,'%');

END IF;

END IF;

END

调用代码

public static List<Person> select\_filter(String sp\_name) throws SQLException

{

List<Person> result = new ArrayList<Person>();

Person p = null;

//1、获得连接

Connection conn = DBUtil.getConnection();

//2、获得callablestatement

CallableStatement cs = conn.prepareCall("call sp\_select\_filter(?)");

cs.setString(1,sp\_name);

//3、执行存储过程

cs.execute();

//4、处理返回的结果:结果集，出参

ResultSet res = cs.getResultSet();

while(res.next())

{

p = new Person();

p.setId(res.getInt("id"));

p.setUser\_name(res.getString("user\_name"));

p.setAge(res.getInt("age"));

result.add(p);

}

return result;

}

JDBC带输出参数存储过程

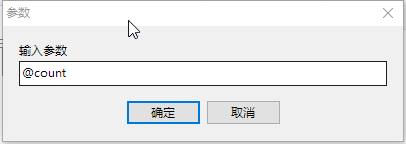
存储过程代码

BEGIN

SELECT count(\*) INTO count FROM person;

END

运行函数



调用代码

public static Integer select\_count() throws SQLException{

Integer count = 0;

//1、获得连接

Connection conn = DBUtil.getConnection();

//2、获得callablestatement

CallableStatement cs = conn.prepareCall("call sp\_select\_count(?)");

cs.registerOutParameter(1, Types.INTEGER);

//3、执行存储过程

cs.execute();

//4、处理返回的结果:结果集，出参

count = cs.getInt(1);

return count;

}

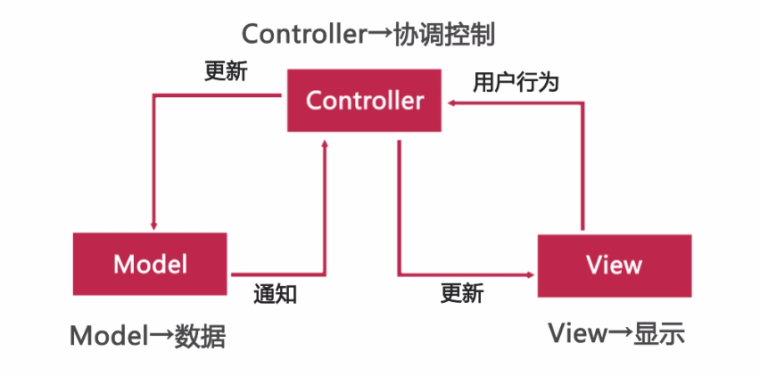
采用MVC三层架构

View 视图层

Control 控制层

Model 模型层

DB 数据库



java.util.Date -> java.sql.Date 需要转换

new Date(p.getBirthday().getTime())

反过来则不需要转换

视图层

流程：

程序启动后，一直保持在运行状态。

循环接收控制台的啊输入参数。

调用Action（控制层）响应，并将返回结果展示在控制台中。

直到输入特定的输入标记（如EXIT）后，程序退出。

问题：

循环接受参数

某个功能的保持

JDBC各种连接方式的对比

1. JDBC+ODBC桥的方式

需要数据库的ODBC驱动

仅适用于微软的系统

1. JDBC+厂商API的形式

厂商API一般用C变那些

1. JDBC+厂商Database Connection Server + DataBase的形式

在JAVA与DATABASE之间架起了一台专门用于数据库

连接的服务器（一般由数据库厂商提供）

1. JDBC+DATABASE的连接方式

这使得Application与数据库分开，开发者只需关心内部

逻辑的实现而不需注重数据库连接的具体实现。

1、数据：描述事务的符号记录。可用文字、图形等多种形式表示，经数字化处理后可存入计算机。

2、数据库（DB）：按一定的数据模型组织、描述和存储在计算机内的、有组织的、可共享的数据集合。

3、数据库管理系统（DBMS）：位于用户和操作系统之间的一层数据管理软件。主要功能包括：

数据定义功能：DBMS提供DDL，用户通过它定义数据对象。

数据操纵功能：DBMS提供DML，用户通过它实现对数据库的查询、插入、删除和修改等操作。

常用的数据模型：层次模型、网状模型、关系模型、面向对象模型

关系模型的基本概念

层次、网状数据库是面向专业人员的，使用很不方便。程序员必须经过良好的培训，对所使用的系统有深入的了解才能用好系统。

关系数据库就是要解决这一问题，使它成为面向用户的系统。

关系数据库是应用数学方法来处理数据的。它具有结构简单、理论基础坚实、数据独立性高以及提供非过程性语言等优点。

关系数据库标准语言——SQL

SQL是 Structured Query Language的缩写。

完成核心功能只有9个动词：

数据查询： SELECT

数据定义： CREATE，DROP，ALTER

数据操纵： INSERT，DELETE，UPDATE

数据控制： GRANT，REVOKE

支持三级模式结构

视图外——模式

基本表（的集合）——模式

存储文件和索引——内模式

基本表的定义和修改

定义：基本格式为

CREATE TABLE 表名（列名1 类型 [列级完整性约束][，列名2 类型 [列级完整性约束]…）；

示例

CREATE TABLE S（ S# CHAR（3）NOT NULL UNIQUE，SN CHAR（15），

SD CHAR（15），SA SMALLINT）；

修改基本表

（1）增加列：

ALTER TABLE 表名 ADD 列名 类型 [完整性约束]；

如 ALTER TABLE S ADD SD INT；

（2）修改列

ALTER TABLE 表名 MODIFY 列名 类型；

如 ALTER TABLE S MODIFY SD CHAR（20）；

（3）删除完整性约束

ALTER TABLE 表名 DROP 完整性约束名；

删除：

DROP TABLE 表名；

CREATE TABLE users(

id SMALL UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(20) NOT NULL,

password VACHAR(32) NOT NULL,

age TINYINT UNSIGNED NOT NULL DEFAULT 10,

sex BOOLEAN

);

INSERT 方法插入记录。

INSERT users VALUE(NULL//自动编号,’TOM’,’123’,25,1);

SELECT \* FROM users;

SHOW COLUMNS FROM users;

INSERT users VALUE(NULL//自动编号,’TOM’,’123’,DEFAULT//默认值,1);

INSERT users VALUE(NULL,’TOM’,’123’,DEFAULT,1),(NULL,’ROSE’,md5(’123’),DEFAULT,0);

//一次性插入多条记录

INSERT SET 方法，此方法可以使用子查询（SubQuery）,一次只能添加一条记录。

INSERT users SET username=’Ben’,password=’456’;

INSERT SELECT 方法，此方法可以将查询结果插入到指定数据表。

使用INSERT SELECT插入记录//将查询的结果写入数据表

SELECT goods\_cates FROM tdb\_goods GROUP BY goods\_cates;//分类

INSERT tdb\_goods\_cates(cate\_name) SELECT goods\_cates FROM tdb\_goods GROUP BY goods\_cates;

将一个表中满足条件的记录，插入另一个记录表。

INSERT test(username) SELECT username FROM users WHERE age>=30;

更新记录（单表更新）

UPDATE users SET age = age + 5;

UPDATE users SET age = age - id ,sex = 0;

UPDATE users SET age = age + 10 WHERE id%2 = 0; //id为偶数

WHERE对记录进行过滤，如果没有指定WHERE子句，则显示所有记录。

在WHERE表达式中，可以使用MySQL支持的函数或运算符。

删除记录（单表删除）

DELETE FROM users WHERE id = 6;

此时重新插入新记录，id号为原有最大id号+1，而不是覆盖被删除的id号。

查找记录

SELECT VERSION();

SELECT NOW();

SELECT 3+5;

SELECT id,username FROM users;

SELECT username,id FROM users;

select查询表达式的顺序会影响结果的顺序。

\*表示所有的列。

SELECT users.id,users.username FROM users;

清晰分辨出字段属于哪个表。

SELECT id AS uid,username AS uname FROM users ;

字段的别名，结果的别名。

查询结果分组

GROUP BY

SELECT sex FROM users GROUP BY sex;

按照性别进行分组。

分组条件

HAVING

SELECT sex,age FROM users GROUP BY sex HAVING age > 35;

SELECT sex FROM users GROUP BY sex HAVING count(id) > = 2;

对查询结果进行排序

ORDER BY

SELECT \* FROM users ORDER BY id DESC ;//降序

SELECT \* FROM users ORDER BY age,id DESC ;//降序

先按age排序如相同，再按id排序。

限制查询结果返回的数量

SELECT \* FROM users LIMIT 2;//第一第二条记录返回

SELECT \* FROM users LIMIT 2,2;//第三第四条记录返回，从0开始数。

SET NAMES GBK;以GBK编码显示数据。

SELECT \* FROM tdb\_goods\G;//以表格的形式输出

子查询

出现在其他SQL语句内的SELECT子句。

返回结果可以返回标量，一行、一列、子查询。

1、比较运算符的子查询

SELECT AVG(goods\_price) FROM tdb\_goods;

SELECT ROUND(AVG(goods\_price),2) FROM tdb\_goods;

对商品价格平均值四舍五入，保留小数点后2位。

SELECT goods\_id,good\_name,goods\_price FROM tdb\_goods WHERE goods\_price >=100;

子查询实现，

SELECT goods\_id,good\_name,goods\_price FROM tdb\_goods WHERE goods\_price >=SELECT ROUND(AVG(goods\_price),2);

用ANY、SOME 、ALL修饰比较运算符。



1. 使用[NOT ] IN 的子查询

多表更新

UPDATE tdb\_goods INNER JOIN tdb\_goods\_cates ON goods\_cate = cate\_name

SET goods\_cate = cate\_id;

参照另外的表来更新本表的参数。

CREATE SELECT

创建数据表并将查询结果写入到数据表

CREATE TABLE tdb\_goods\_brands

(

brand\_id SMALL UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

brand\_name VARCHAR(40) NOT NULL

)

SELECT brand\_name FROM tdb\_goods GROUP BY brand\_name;

内连接 INNER JOIN 显示左表和右表符合连接条件的记录

左外连接 显示左表的全部记录及右表符合连接的记录

右外连接 显示右表的全部记录及左表符合连接的记录

多表更新 多表删除

运算符与函数

字符函数

CONCAT() 将字符进行连接

CONCAT\_WS() 以指定的分隔符进行连接

SELECT CONCAT(‘My’,’SQL’);

FORMAT() 数字格式化

LOWER() 小写字符

UPPER() 大写字符

LEFT() 获取左侧字符

RIGHT() 获取右侧字符

LENGTH() 获取字符长度

LTRIM() 删除前导空格

RTRIM() 删除后续空格

TRIM() 删除前后空格

SELECT TRIM(LEADING ‘?’ FROM ‘??MySQL???’);

删除前导问号

SELECT TRIM(TRAILING ‘?’ FROM ‘??MySQL???’);

删除后续问号

SELECT TRIM(BOSS‘?’ FROM ‘??MySQL???’);

删除前后问号

REPLACE() 替换

SUBSTRING() 字符串截取

SELECT SUBSTRING (‘MySQL’,1,2)； 从第一个截取，截2位。

[NOT] LIKE 模式匹配

SELECT ‘MySQL’ LIKE ‘M%’；

SELECT \* FROM test WHERE name LIKE ‘%1%%’ ESCAPE ‘1’;

数值运算符函数

SELECT CEIL(3.01) ==>4 /\*有n.xx 都是n+1 进一取整 向上取整

SELECT FLOOR(3.99) ==>3 /\*取n.xx 都是n 舍一取整 向下取整

SELECT 3 DIV 4 ==> 0 /\*整数除法，取商

SELECT 3/4 ==>0.75 /\*除法

SELECT 2.1 MOD 2 ==>0.1 /\*取余数(取模) 被除数是整数,小数都可以

SELECT 21 % 2 ==>1 /\*取余数 mod=%

SELECT POWER(3,4) ==>81 /\*3的4次方 幂运算

SELECT ROUND(3.1415926,4) ==>3.1416 /\*四舍五入，保留4位小数

SELECT TRUNCATE(123.89,1) ==>123.8 /\*截取小数点后的位数

SELECT TRUNCATE(123.89,0) ==>123 /\*截取小数点后的位置,0位为整数部分

SELECT TRUNCATE(123.89,-1) ==>120 /\*截取-1,从个位起去掉后面的数值替换为0

比较运算符和函数

[NOT]BETWEEN…AND…[不]在范围之内，结果返回1或0。

例如：15 BETWEEN 10 AND 16; //1。

[NOT]IN()列出值选项。

例如：SELECT 10 IN(5,10,25); //1。

IS [NOT] NULL：只有NULL IS [NOT] NULL返回1，其他情况（包括”,0）都返回0。

日期时间函数

（1）SELECT NOW(); // 当前日期时间。

（2）SELECT CURDATE(); // 当前日期。

（3）SELECT CURTIME(); // 当前时间。

（4）SELECT DATE\_ADD(‘2014-3-12’,INTERVAL [-]365 DAY); // 2015-03-12，DAY还可以替换为WEEK，MONTH，YEAR等。DATE\_ADD不是增加的意思，是变化，既可以增加，也可以减去。

（5）DATEDIFF(); // 日期的差值。

DATEDIFF(‘2013-3-12’,’2014-3-12’); // -365 日期差值。

（6）DATE\_FORMAT(); // 日期格式化。

SELECT DATE\_FORMAT(‘2014-3-2’,’%m/%d/%Y’); // 03/02/2014（’%’带有前导0作用）。

信息函数

SELECT CONNECTION\_ID(); ==>2 /\*连接ID

SELECT DATABASE(); /\*当前打开的数据库

SELECT LAST\_INSERT\_ID(); /\*最后插入的记录 同时写入多条记录是只返回写入的第一条的id。而且表中必须要有自动编号的字段。

DESC tab\_name; /\*查看数据库字段及属性，查看表结构

SELECT USER(); /\* 打印当前用户信息

SELECT VERSION(); /\*打印当前数据库信息

聚合函数

只有一个返回值

AVG() - 平均值

SELECT ROUND(AVG(goods\_price),2) AS avg\_price FROM tdb\_goods;

COUNT() - 计数

SELECT COUNT(goods\_id) as counts FROM tdb\_goods;

MAX() - 最大值

SELECT MAX(goods\_price) as counts FROM tdb\_goods;

MIN() - 最小值

SUM() - 求和

SELECT SUM(goods\_price) as counts FROM tdb\_goods;

加密函数

MD5()：信息摘要算法

select md5（’admin’）;//给admin加密。如果mysql中的信息是为了web开发做准备，那么尽量使用md5

password（）：密码算法

set password=password（‘1236’）修改了当前登录密码为1236

创建自定义函数

CREATE FUNCTION function\_name(param1, param2)

RETURNS 返回值类型{STRING|INTEGER|REAL|DECIMAL}

routine\_body - 函数体

CREATE FUNCTION SHOW\_AVG(num1 SMALLINT UNSIGNED , num2 SMALLINT UNSIGNED)

RETURNS FLOAT(10,2) UNSIGNED

RETURN (num1 + num2)/2;

删除函数：DROP FUNCTION function\_name;

查询函数：SHOW FUNCTION STATUS;

调用函数：SELECT f2(10,15);

修改分隔符：DELMITER 分隔符

例：DELIMITER //

/\* 将分隔符修改为 ‘//’

当函数体内需要执行的是多条语句时，要使用BEGIN…END语句，

且当编写函数体内容的时候，需要使用 DELIMITER 关键字将分隔符先修改为别的，否则编写语句的时候写到 ‘;’ 的时候会直接执行，导致函数编写失败，

删除函数：DROP FUNCTION [IF EXISTS] function\_name

\*/

例：

DELIMITER //

CREATE FUNCTION ADD\_USER(p\_id SMALLINT,username VARCHAR(20))

RETURNS INT UNSIGNED

BEGIN

INSERT user(p\_id,username) VALUES(p\_id,username);

RETURN LAST\_INSERT\_ID();

END

1、MySQL执行流程，通过存储过程，可以避免重复语法分和编译，提高执行效率

2、存储过程：

（1）是SQL语句与控制语句的【预编译集合】，以【一个名称存储】作为【一个单元处理】

（2）优点：

· 增强了语句的功能和灵活性：可以通过控制语句对流程进行控制和判断

· 实现较快的执行速度，只在【第一次调用时进行语法分析和编译】 ，以后直接从内存中得到结果

· 减少网络流量

存储过程语法结构解析

1.创建存储过程

CREATE

[DEFINER = {userCURRENT\_USER}]

PROCEDURE sp\_name ([proc\_parameter[,…]]) //可以带0到多个参数

[characteristic …] routine\_body

其中参数

proc\_parameter:

[IN OUT INOUT] param\_name type

ps:

IN, 表示该参数的值必须在调用存储过程时指定

OUT, 表示该参数的值可以被存储过程改变，并且可以返回

INOUT, 表示该参数的值调用时指定，并且可以被改变和返回

2.特性

COMMENT ‘string’

{CONTAINS SQL NO SQL READS SQL DATA MODIFIES SQL DATA SQL SECURITY{DEFINER INVOKER}

ps:

COMMENT:注释

CONTAINS SQL:包含SQL语句, 但不包含读或写数据的语句

NO SQL:不包含SQL语句

READS SQL DATA:包含读数据的语句

MODIFIES SQL DATA:包含写数据的语句

SQL SECURITY {DEFINERINVOKER}指明谁有权限来执行

3.过程体

(1)过程体由合法的SQL语句构成;

(2)过程体可以是任意SQL语句;(增删改查)

(3)过程体如果为复合结构则使用BEGIN…END语句

(4)复合结构可以使用条件、循环等控制语句

使用存储过程

创建没有参数的存储过程

CREATE PROCEDURE sp1() SELECT VERSION();

/创建实例代码（参数名不能和表中的字段名相同）/

创建带有INT类型参数的存储过程：

DELIMITER // //定界符

CREATE PROCEDURE removeUserById(IN p\_id INT UNSIGNED) //int 是数据类型

BEGIN DELETE FROM users WHERE id = p\_id;

END

// //结束

DELIMITER ;

创建IN OUT参数的存储过程

DELIMITER //

CREATE PROCEDURE removerUserAndReturnUserName(IN showID INT UNSIGNED,OUT showName INT UNSIGNED)

BEGIN

DELETE FROM user WHERE id = showID;

SELECT count(ID) FROM user INTO showName;

END

//

SELECT count(ID) FROM user INTO showName;

/\* 该语句中的 INTO 含义就是将 SELECT 语句结果的表达式返回到 showName 变量中 /

CALL removerUserAndReturnUserName(10,@nums); /\* @nums 所代表的就是用户变量，可用 SELECT @nums 输出 /

用 DECLARE 声明的变量是局部变量，局部变量只能存在于 BEGIN…END 之间，且声明时必须置于 BEGIN…END 的第一行

而通过 SELECT…INTO…/SET @id = 07 这种方法设置的变量我们称之为用户变量，只能存在于当前用户所使用的客户端有效。

CALL rmUserAndRtUserNums(27, @nums);

SELECT @nums; //@nums - 就是用户变量

创建带有多个OUT类型参数的存储过程

INSERT test(username) VALUES(‘A’),(‘B,’),(‘C’);

SELECT ROW\_COUNT(); //返回被插入的记录总数

SELECT \* FROM test;

UPDATE test SET username = CONTCAT(username, ‘–immoc’) WHERE id <= 2;

SELECT row\_COUNT(); //返回更新的记录总数

DELIMITER //

CREATE PROCEDURE rmUserByAgeAndRtInfos(IN p\_age SMALLINT UNSIGNED, OUT delNums SMALLINT UNSIGNED, OUT leftNums SMALLINT UNSIGNED)

BEGIN

DELETE FROM users WHERE age = p\_age; //注意变量不同

SELECT ROW\_COUNT() INTO delNums; //返回插入、删除和更新的受影响的记录总数

SELECT COUNT(id) FROM users INTO leftNums;

END

修改存储过程：(修改注释，名称，不能修改过程体)

ALTER PROCEDURE sp\_name [characteristic]

COMMENT ‘string’

{CONTAINS SQLNO SQLREADS SQL DATAMODIFIES SQL DATA}

SQL SECURITY{DEFINERINVOKER}

删除存储过程：

DROP PROCEDURE [IF EXISTS] sp\_name;

调用存储过程

CALL sp\_name([parameter[,…]]) //带参数的存储过程的调用

CALL sp\_name[()] //不带参数的存储过程调用

CALL sp1; or CALL sp1();