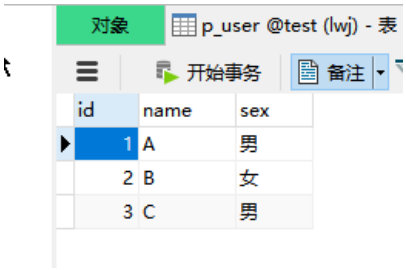
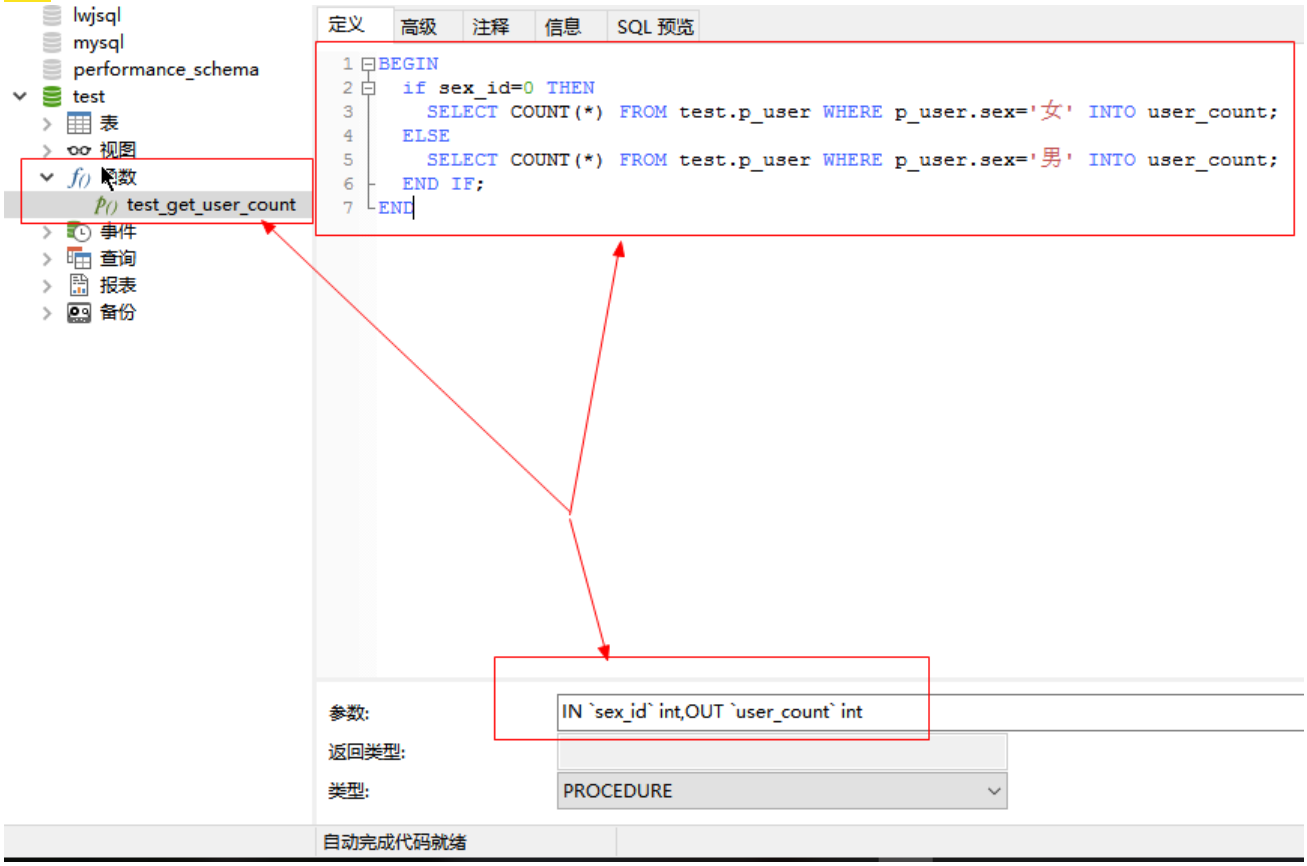
调用存储过程

1.创建表



2.新建存储过程

```
BEGIN
if sex_id=0 THEN
 SELECT COUNT(*) FROM test.p_user WHERE p_user.sex='女' INTO user_count;
ELSE
 SELECT COUNT(*) FROM test.p_user WHERE p_user.sex='男' INTO user_count;
END IF;
END
```

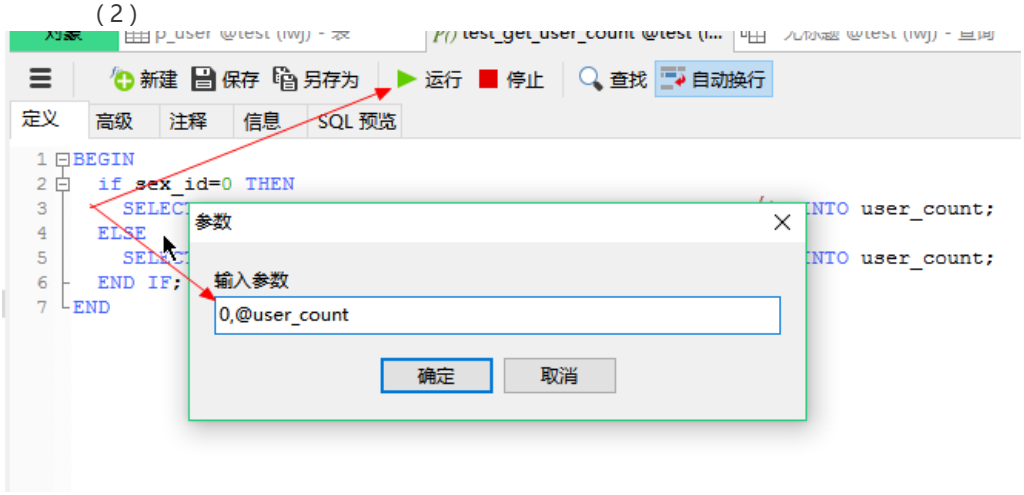


这个存储过程运行的方法有两种

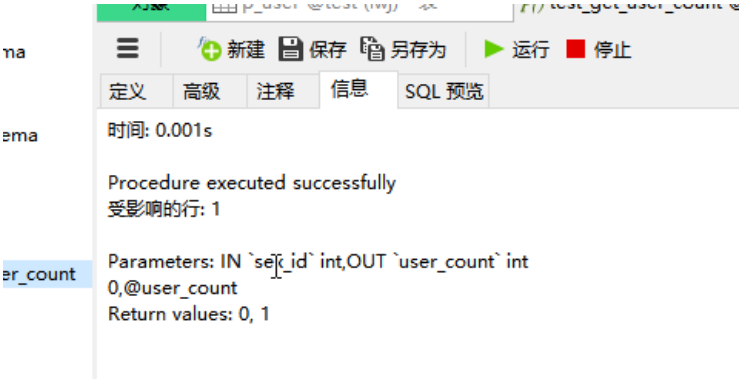
（1）

```
SET @user_count=0;
CALL test.test_get_user_count(0,@user_count);
SELECT @user_count;
```

（2）



得到的结果是



这样在数据库中试过可以调用存储过程啦，下面是在应用程序中调用~

3.创建的mapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mccken.test8.UserMapper">

 <select id="getUserCount" parameterMap="getUserCountMap" statementType="CALLABLE">
  CALL test_get_user_count(?,?)
 </select>

 <parameterMap type="java.util.Map" id="getUserCountMap">
  <parameter property="sexid" mode="IN" jdbcType="INTEGER"/>
  <parameter property="usercount" mode="OUT" jdbcType="INTEGER"/>
```

</parameterMap>

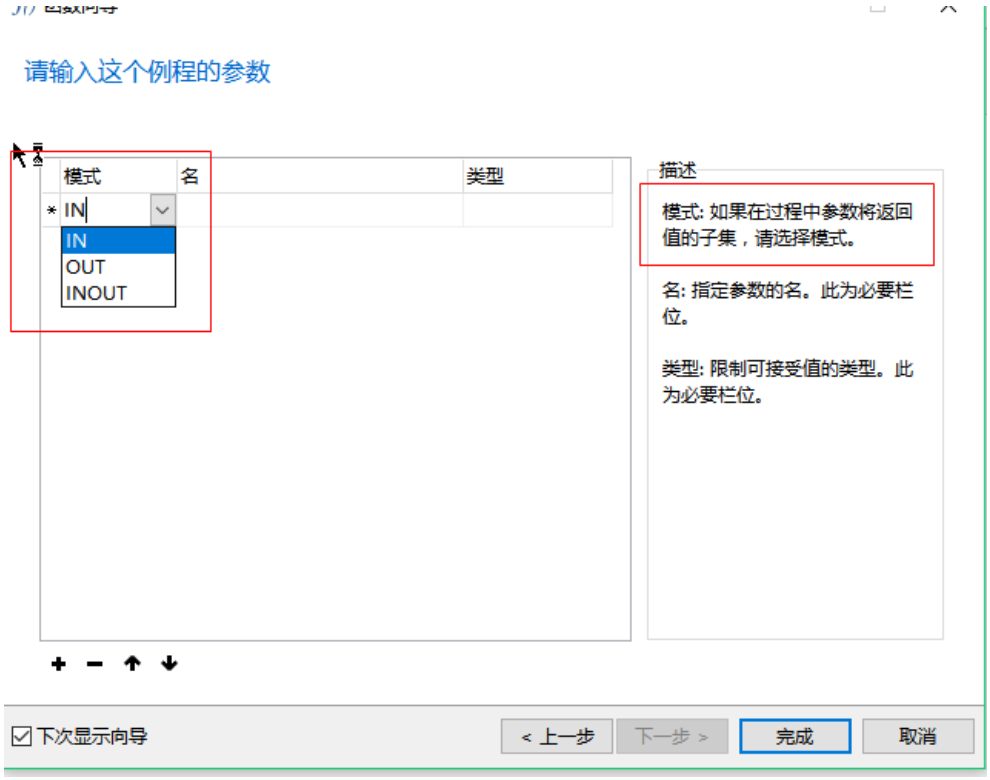
</mapper>

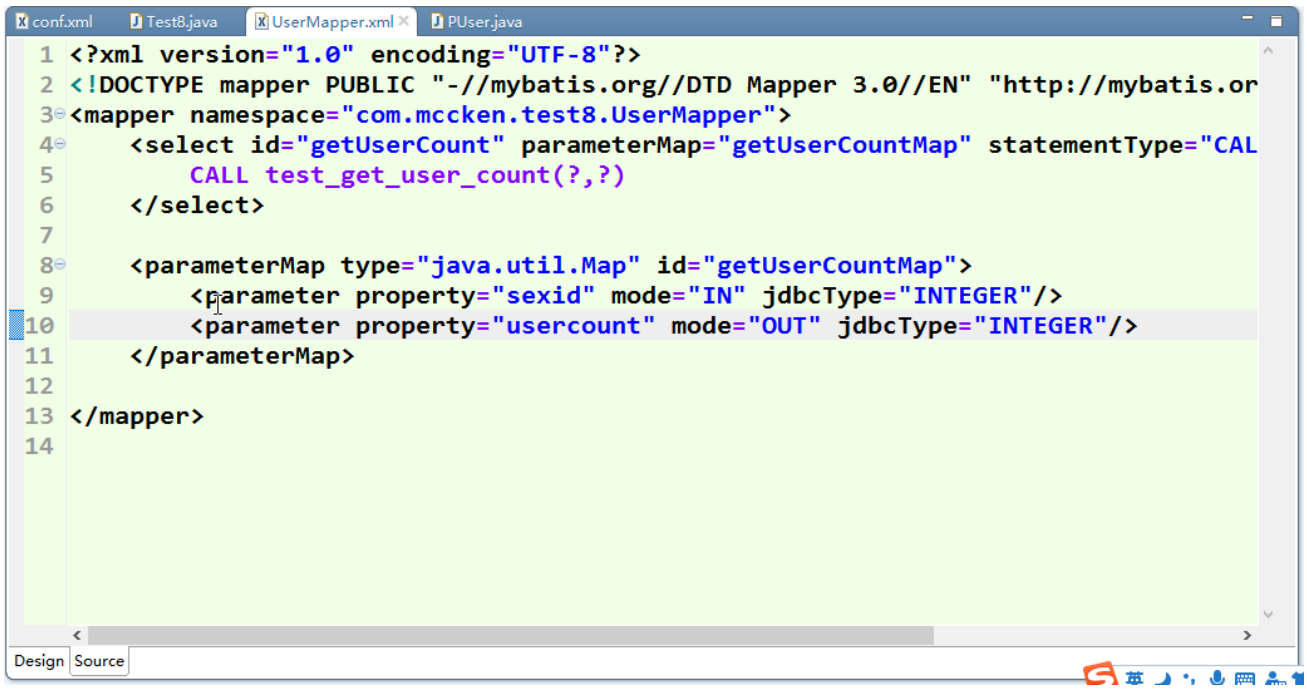
****

<select>标签体的内容是调用存储过程

CALL test_get_user_count(?,?)

引用了parameterMap属性，在parameterMap标签中的Type为java.util.Map（因为应用程序里用的是map传值），传来的map中有两个参数，分别为sexid,usercount。不同的是sexid为IN模式（mode=IN）,而usercount为OUT模式（mode=OUT）,另外还要设置jdbcType为传入的值的类型。最后还要在<select>里设置属性statementType="CALLABLE"





4.创建java程序

```
    @Test
    public void getClasses(){
 SqlSessionFactory factory = MybatisUtils.getFactory();
 SqlSession session = factory.openSession();

 String statement="com.mccken.test8.UserMapper.getUserCount";

        Map<String,Integer> map = new HashMap<String,Integer>();

 map.put("sexid", 0);
 map.put("usercount", -1);
 session.selectOne(statement, map);
 Integer integer = map.get("usercount");
 System.out.println(integer);
 session.close();
    }
```

****

创建HashMap以用来传值

```
Map<String,Integer> map = new HashMap<String,Integer>();
```

将sexid和usercount传入map中

```
        map.put("sexid", 0);
        map.put("usercount", -1);
```

调用session.select，将map传到mapper.xml中，并返回mode=OUT的usercount的值

```
session.selectOne(statement, map);
Integer integer = map.get("usercount");
```

```
public class Test8 {
    @Test
    public void getClasses(){
        SqlSessionFactory factory = MybatisUtils.getFactory();
        SqlSession session = factory.openSession();

        String statement="com.mccken.test8.UserMapper.getUserCount";
        Map<String,Integer> map = new HashMap<String,Integer>();

        map.put("sexid", 0);
        map.put("usercount", -1);
        session.selectOne(statement, map);
        Integer integer = map.get("usercount");
        System.out.println(integer);


        session.close();
    }
}
```

**********以下是关于存储过程的内容**********

**********以下是关于存储过程的内容**********
**********以下是关于存储过程的内容**********
**********以下是关于存储过程的内容**********

http://blog.sina.com.cn/s/blog_52d20fbf0100ofd5.html

1. 存储过程简介

我们常用的操作数据库语言SQL语句在执行的时候需要要先编译，然后执行，而存储过程（Stored Procedure）是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行它。

2.好处

存储过程在数据库中被创建后，可以在程序中被多次调用，而不必重新编写该存储过程的SQL语句

3.MySQL存储过程的创建

(1). 格式

MySQL存储过程创建的格式：CREATE PROCEDURE 过程名 ([过程参数[,...]])
[特性 ...] 过程体

这里先举个例子：

```
1. mysql> DELIMITER //
2. mysql> CREATE PROCEDURE proc1(OUT s int)
3.     -> BEGIN
4.     -> SELECT COUNT(*) INTO s FROM user;
5.     -> END
6.     -> //
7. mysql> DELIMITER ;
```

注：
（1）这里需要注意的是DELIMITER //和DELIMITER ;两句，DELIMITER是分割符的意思，因为MySQL默认以";"为分隔符，如果我们没有声明分割符，那么编译器会把存储过程当成SQL语句进行处理，则存储过程的编译过程会报错，所以要事先用DELIMITER关键字申明当前段分隔符，这样MySQL才会将";"当做存储过程中的代码，不会执行这些代码，用完了之后要把分隔符还原。
（2）存储过程根据需要可能会有输入、输出、输入输出参数，这里有一个输出参数s，类型是int型，如果有多个参数用","分割开。
（3）过程体的开始与结束使用BEGIN与END进行标识。

MySQL存储过程详解 mysql 存储过程

王者佳暮 2010-12-13 10:59:38

mysql存储过程详解
1. 存储过程简介

我们常用的操作数据库语言SQL语句在执行的时候

我们常用的操作数据库语言SQL语句在执行的时候需要要先编译，然后执行，而存储过程（Stored Procedure）是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行它。

一个存储过程是一个可编程的函数，它在数据库中创建并保存。它可以有SQL语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。数据库中的存储过程可以看做是对编程中面向对象方法的模拟。它允许控制数据的访问方式。

存储过程通常有以下优点：

(1).存储过程增强了SQL语言的功能和灵活性。存储过程可以用流控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算。

(2).存储过程允许标准组件是编程。存储过程被创建后，可以在程序中被多次调用，而不必重新编写该存储过程的SQL语句。而且数据库专业人员可以随时对存储过程进行修改，对应用程序源代码毫无影响。

(3).存储过程能实现较快的执行速度。如果某一操作包含大量的Transaction-SQL代码或分别被多次执行，那么存储过程要比批处理的执行速度快很多。因为存储过程是预编译的。在首次运行一个存储过程时查询，优化器对其进行分析优化，并且给出最终被存储在系统表中的执行计划。而批处理的Transaction-SQL语句在每次运行时都要进行编译和优化，速度相对要慢一些。

(4).存储过程能过减少网络流量。针对同一个数据库对象的操作（如查询、修改），如果这一操作所涉及的Transaction-SQL语句被组织程存储过程，那么当在客户计算机上调用该存储过程时，网络中传送的只是该调用语句，从而大大增加了网络流量并降低了网络负载。

(5).存储过程可被作为一种安全机制来充分利用。系统管理员通过执行某一存储过程的权限进行限制，能够实现对相应的数据的访问权限的限制，避免了非授权用户对数据的访问，保证了数据的安全。

2. 关于MySQL的存储过程

存储过程是数据库存储的一个重要的功能，但是MySQL在5.0以前并不支持存储过程，这使得MySQL在应用上大打折扣。好在MySQL 5.0终于开始已经支持存储过程，这样即可以大大提高数据库的处理速度，同时也可以提高数据库编程的灵活性。

3. MySQL存储过程的创建

(1). 格式

MySQL存储过程创建的格式：CREATE PROCEDURE 过程名 ([过程参数[,...]])
[特性 ...] 过程体

这里先举个例子：

```
1. mysql> DELIMITER //
2. mysql> CREATE PROCEDURE proc1(OUT s int)
3.     -> BEGIN
4.     -> SELECT COUNT(*) INTO s FROM user;
5.     -> END
6.     -> //
7. mysql> DELIMITER ;
```

注：

（1）这里需要注意的是DELIMITER //和DELIMITER ;两句，DELIMITER是分割符的意思，因为MySQL默认以";"为分隔符，如果我们没有声明分割符，那么编译器会把存储过程当成SQL语句进行处理，则存储过程的编译过程会报错，所以要事先用DELIMITER关键字申明当前段分隔符，这样MySQL才会将";"当做存储过程中的代码，不会执行这些代码，用完了之后要把分隔符还原。

（2）存储过程根据需要可能会有输入、输出、输入输出参数，这里有一个输出参数s，类型是int型，如果有多个参数用","分割开。

（3）过程体的开始与结束使用BEGIN与END进行标识。

这样，我们的一个MySQL存储过程就完成了，是不是很容易呢?看不懂也没关系，接下来，我们详细的讲解。

(2). 声明分割符

其实，关于声明分割符，上面的注解已经写得很清楚，不需要多说，只是稍微要注意一点的是：如果是用MySQL的Administrator管理工具时，可以直接创建，不再需要声明。

(3). 参数

MySQL存储过程的参数用在存储过程的定义，共有三种参数类型,IN,OUT,INOUT,形式如：

CREATE PROCEDURE([[IN |OUT |INOUT] 参数名 数据类形...])

IN 输入参数:表示该参数的值必须在调用存储过程时指定，在存储过程中修改该参数的值不能被返回，为默认值

OUT 输出参数:该值可在存储过程内部被改变，并可返回

INOUT 输入输出参数:调用时指定，并且可被改变和返回

Ⅰ. IN参数例子

创建:

```
1. mysql > DELIMITER //
2. mysql > CREATE PROCEDURE demo_in_paramet
3. -> BEGIN
4. -> SELECT p_in;
5. -> SET p_in=2;
6. -> SELECT p_in;
7. -> END;
8. -> //
9. mysql > DELIMITER ;
```

执行结果:

```
1. mysql > SET @p_in=1;
```

```
mysql > CALL demo_in_parameter(@p_in);
+------+
| p_in |
+------+
|   1  |
+------+

+------+
| p_in |
+------+
|   2  |
+------+

mysql> SELECT @p_in;
+-------+
| @p_in |
+-------+
|  1    |
+-------+
```

以上可以看出，p_in**虽然在存储过程中被修改，但并不影响**@p_id**的值**

Ⅱ.OUT参数例子
创建:

```
mysql > DELIMITER //
mysql > CREATE PROCEDURE demo_out_parame

-> BEGIN
-> SELECT p_out;
-> SET p_out=2;
-> SELECT p_out;
-> END;
-> //
mysql > DELIMITER ;
```

执行结果:

```
mysql > SET @p_out=1;
mysql > CALL sp_demo_out_parameter(@p_ou

+-------+
| p_out |
+-------+
| NULL  |
+-------+

+-------+
| p_out |
+-------+
|   2   |
+-------+

mysql> SELECT @p_out;
+-------+
```

```
| p_out |
+-------+
|   2   |
+-------+
```

Ⅲ. INOUT参数例子

创建:

```
mysql > DELIMITER //
mysql > CREATE PROCEDURE demo_inout_paran

-> BEGIN
-> SELECT p_inout;
-> SET p_inout=2;
-> SELECT p_inout;
-> END;
-> //
mysql > DELIMITER ;
```

执行结果:

```
mysql > SET @p_inout=1;
mysql > CALL demo_inout_parameter(@p_inout

+----------+
| p_inout |
+----------+
|    1    |
+----------+

+----------+
| p_inout |
+----------+
|    2    |
+----------+

mysql > SELECT @p_inout;
+-----------+
| @p_inout |
+-----------+
|    2     |
+-----------+
```

(4). 变量

Ⅰ. 变量定义

DECLARE variable_name [,variable_name...] datatype [DEFAULT value];

其中， datatype**为**MySQL**的数据类型，如**:int, float, date, varchar(length)

例如:

```
DECLARE l_int int unsigned default 4000000;
DECLARE l_numeric number(8,2) DEFAULT 9.95;

DECLARE l_date date DEFAULT '1999-12-31';
DECLARE l_datetime datetime DEFAULT '1999-
```

```
    12-31 23:59:59';
5. DECLARE l_varchar varchar(255) DEFAULT 'This w
```

Ⅱ. 变量赋值

SET **变量名** = **表达式值** [,variable_name = expression ...]

Ⅲ. 用户变量

ⅰ. 在MySQL客户端使用用户变量

```
1. mysql > SELECT 'Hello World' into @x;
2. mysql > SELECT @x;
3. +-------------+
4. |   @x        |
5. +-------------+
6. | Hello World |
7. +-------------+
8. mysql > SET @y='Goodbye Cruel World';
9. mysql > SELECT @y;
10. +---------------------+
11. |     @y              |
12. +---------------------+
13. | Goodbye Cruel World |
14. +---------------------+
15.
16. mysql > SET @z=1+2+3;
17. mysql > SELECT @z;
18. +------+
19. | @z   |
20. +------+
21. |  6   |
22. +------+
```

ⅱ. 在存储过程中使用用户变量

```
1. mysql > CREATE PROCEDURE GreetWorld( ) SELI

2. mysql > SET @greeting='Hello';
3. mysql > CALL GreetWorld( );
4. +----------------------------+
5. | CONCAT(@greeting,' World') |
6. +----------------------------+
7. |  Hello World               |
8. +----------------------------+
```

ⅲ. 在存储过程间传递全局范围的用户变量

```
1. mysql> CREATE PROCEDURE p1()   SET @last_p

2. mysql> CREATE PROCEDURE p2() SELECT CONC

3. mysql> CALL p1( );
4. mysql> CALL p2( );
5. +-------------------------------------------+
6. | CONCAT('Last procedure was ',@last_proc   |
```

```
+---------------------------------------------+
| Last procedure was p1                       |
+---------------------------------------------+
```

注意:

①用户变量名一般以@开头

②滥用用户变量会导致程序难以理解及管理

(5). 注释

MySQL存储过程可使用两种风格的注释

双模杠：--

该风格一般用于单行注释

c风格： 一般用于多行注释

例如：

```
mysql > DELIMITER //
mysql > CREATE PROCEDURE proc1 --name存储过程名
-> (IN parameter1 INTEGER)
-> BEGIN
-> DECLARE variable1 CHAR(10);
-> IF parameter1 = 17 THEN
-> SET variable1 = 'birds';
-> ELSE
-> SET variable1 = 'beasts';
-> END IF;
-> INSERT INTO table1 VALUES (variable1);
-> END
-> //
mysql > DELIMITER ;
```

4. MySQL存储过程的调用

用call和你过程名以及一个括号，括号里面根据需要，加入参数，参数包括输入参数、输出参数、输入输出参数。具体的调用方法可以参看上面的例子。

5. MySQL存储过程的查询

我们像知道一个数据库下面有那些表，我们一般采用show tables;进行查看。那么我们要查看某个数据库下面的存储过程，是否也可以采用呢？答案是，我们可以查看某个数据库下面的存储过程，但是是令一钟方式。

我们可以用

select name from mysql.proc where db='数据库名';

或者

select routine_name from information_schema.routines where routine_schema='数据库名';

或者

show procedure status where db='数据库名';

进行查询。

如果我们想知道，某个存储过程的详细，那我们又该怎么做呢？是不是也可以像操作表一样用describe 表名进行查看呢？

答案是，我们可以查看存储过程的详细，但是需要

结果是：我们可以查看存储过程的详细，但是需要用另一种方法：

SHOW CREATE PROCEDURE 数据库.存储过程名;

就可以查看当前存储过程的详细。

6. MySQL存储过程的修改

ALTER PROCEDURE

更改用CREATE PROCEDURE 建立的预先指定的存储过程，其不会影响相关存储过程或存储功能。

7. MySQL存储过程的删除

删除一个存储过程比较简单，和删除表一样：

DROP PROCEDURE

从MySQL的表格中删除一个或多个存储过程。

8. MySQL存储过程的控制语句

(1). 变量作用域

内部的变量在其作用域范围内享有更高的优先权，当执行到end。变量时，内部变量消失，此时已经在其作用域外，变量不再可见了，应为在存储
过程外再也不能找到这个申明的变量，但是你可以通过out参数或者将其值指派
给会话变量来保存其值。

```
1. mysql > DELIMITER //
2. mysql > CREATE PROCEDURE proc3()
3.       -> begin
```