## Spring MVC

### 1.@RequestBody User user 注解接收json数据报415（POST请求）

解决办法：

springmvc-servlet.xml 添加

<mvc:annotation-driven/>  
<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">  
 <property name="messageConverters">  
 <list>  
 <ref bean="jsonHttpMessageConverter" />  
 </list>  
 </property>  
</bean>  
<bean id="jsonHttpMessageConverter" class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>application/json;charset=UTF-8</value>  
 </list>  
 </property>  
</bean>

pom.xml 添加

<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-core</artifactId>  
 <version>2.5.2</version>  
</dependency>  
<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.5.2</version>  
</dependency>

备注：使用的是Spring4.3.2(这个很重要)

## MySQL 5.6

### 前言：

**存储函数与存储过程的区别**

一、 存储函数有且只有一个返回值，而存储过程不能有返回值。

二、 函数只能有输入参数，而且不能带in, 而存储过程可以有多个in,out,inout参数。

三、 存储过程中的语句功能更强大，存储过程可以实现很复杂的业务逻辑，而函数有很多限制，如不能在函数中使用insert,update,delete,create等语句；存储函数只完成查询的工作，可接受输入参数并返回一个结果，也就是函数实现的功能针对性比较强。

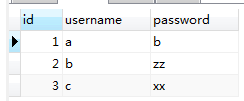
四、 存储过程可以调用存储函数。但函数不能调用存储过程。

五、 存储过程一般是作为一个独立的部分来执行(call调用)。而函数可以作为查询语句的一个部分来调用。

### 1.储存过程（输出参数）

-- MySQL 5.6

-- user表数据



示例1：

-- 新建储存过程(有输出值)

CREATE PROCEDURE abc(out s int) -- out s int 输出

begin

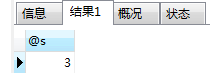
SELECT COUNT(1) into s from user; -- 将查询结果注入到s中

end

call abc(@s); -- 回调储存过程



SELECT @s; -- 获取储存过程的结果



示例2：

-- 新建储存过程(有输出值)

CREATE PROCEDURE abc(out s int) -- out s int 输出

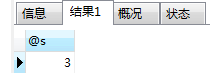
begin

SELECT COUNT(1) into s from user; -- 将查询结果注入到s中

SELECT s;

end

call abc(@s); -- 回调储存过程



### 2.储存过程（输入参数）

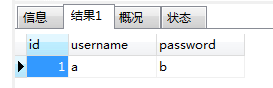
CREATE PROCEDURE cba(in s int) -- in s int 输入参数

begin

SELECT \* from user where id = s;

end

call cba(1)； -- 回调储存过程



### 储存函数

注意：存储函数有且只有一个返回值

存储函数只支持输入参数

-- 存储函数

CREATE FUNCTION fun\_abc(x int)

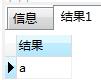
RETURNS VARCHAR(20)

BEGIN

RETURN (SELECT username from user where id = x);

END

SELECT fun\_abc(1) as 结果; -- 调用函数



### 4. if 的使用 ， case when then 的使用

-- 示例1

SELECT

CASE username

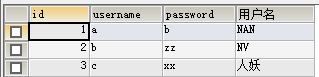
WHEN 'a' THEN 'NAN'

WHEN 'b' THEN 'NV'

ELSE '人妖'

END AS 用户名

FROM USER



-- 示例2

SELECT \*,

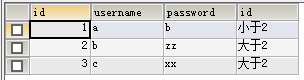
CASE id

WHEN id<2 THEN '小于2'

ELSE '大于2'

END AS i

FROM USER

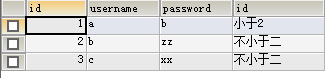


-- 示例3

SELECT \*,

IF(id < 2 , '小于2' , '不小于二') AS id

FROM USER



-- 示例4

create procedure text(out rtn int, in LoginId int)

begin

set rtn=1;

IF LoginId = 3

THEN

set rtn=2;

ELSEIF LoginId = 0

THEN

set rtn=3;

ELSE

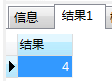
set rtn=4;

END IF;

SELECT rtn as 结果;

end

call text(@aa, 1); -- 调用回调过程



### 5.游标

CREATE PROCEDURE useCursor() -- 创建存储过程

BEGIN

DECLARE no\_more\_record INT DEFAULT 0; -- 声明变量

declare x int; -- 声明变量

DECLARE curl CURSOR FOR SELECT id FROM user; -- 声明游标

DECLARE CONTINUE HANDLER FOR NOT FOUND SET no\_more\_record = 1; -- 报错，光标没找到设置no\_more\_record为1

OPEN curl; -- 打开游标

FETCH curl INTO x; -- 从游标取出一个的数据注入到x中

WHILE no\_more\_record != 1 DO -- 退出条件

if x > 1 then UPDATE user set username = CONCAT(username,x,'1') where id = x;

else UPDATE user set username = CONCAT(username,x,'2') where id = x;

end if;

FETCH curl INTO x; -- 从游标取出一个的数据注入到x中

END WHILE; -- 结束循环

CLOSE curl; -- 关闭游标

END

call useCursor(); -- 调用存储过程

