# 1、in out

<http://blog.csdn.net/paopaomm/article/details/51851919>

本人要调用的存储过程有返回值，如下：

**[sql]** [view plain](http://blog.csdn.net/paopaomm/article/details/51851919) [copy](http://blog.csdn.net/paopaomm/article/details/51851919)

1. **create** or replace **procedure** p\_bakuplog(flag in **out** varchar2)
2. **is**
3. ----------------------------------------------------------------------------------------------
4. -- 作    者:mengpaopao
5. -- 日    期:2016/07/07
6. -- 功能描述:备份清理日志
7. -- 参数注释:
8. -- 返 回 值：flag 0：正常；1：未做操作；2：异常
9. -- 备    注:
10. -- 修改历史:
11. -- [时    间]                 内容                           修改人
12. ----------------------------------------------------------------------------------------------
13. ----------------------------------------------------------------------------------------------
14. type v\_collect **is** **table** **of** varchar2(300) **index** **by** pls\_integer;
15. v\_backup\_time varchar2(8):= to\_char(sysdate,'yyyymmdd');
16. v\_tab v\_tables:=v\_tables(
17. 'ENTRY\_LOG',
18. 'JOBCHAN\_LOG',
19. 'JOB\_LOG',
20. 'STEP\_LOG',
21. 'TRANS\_LOG'
22. );
23. **begin**
25. <pre **name**="code" class="sql">  略...

end;

## 1、首先需要依附一个表定义存储过程；

## @NamedStoredProcedureQuery

## @StoredProcedureParameter

**[java]** [view plain](http://blog.csdn.net/paopaomm/article/details/51851919) [copy](http://blog.csdn.net/paopaomm/article/details/51851919)

1. **package** com.xxxx.core.main.etl.bean;
2. **import** java.util.Date;
3. **import** javax.persistence.\*;
4. /\*\*
5. \*JOBCHAN日志表
6. \*   @AUTO
7. \*/
8. @Entity(name = "jobchan\_log")
9. @NamedStoredProcedureQuery(name = "p\_bakuplog", procedureName = "p\_bakuplog", parameters = {
10. @StoredProcedureParameter(mode = ParameterMode.INOUT, name = "flag", type = String.**class**)
11. })
12. **public** **class** JobchanLog {
14. // 作业批次id
15. @Id
16. @Column(name="id\_batch",updatable=**false**)
17. **private** Integer idBatch;
19. // 日志频道id
20. @Column(name="channel\_id")
21. **private** String channelId;
23. // 父jobid
24. @Column(name="log\_date")
25. **private** Date logDate;
27. // job名称
28. @Column(name="logging\_object\_type")
29. **private** String loggingObjectType;
31. // 读取行数
32. @Column(name="object\_name")
33. **private** String objectName;
35. // 写入行数
36. @Column(name="object\_copy")
37. **private** String objectCopy;
39. // 更新行数
40. @Column(name="repository\_directory")
41. **private** String repositoryDirectory;
43. // 输入行数
44. @Column(name="filename")
45. **private** String filename;
47. // 输出行数
48. @Column(name="object\_id")
49. **private** String objectId;
51. // 拒绝行数
52. @Column(name="object\_revision")
53. **private** String objectRevision;
55. // 错误
56. @Column(name="parent\_channel\_id")
57. **private** String parentChannelId;
59. // 结果集
60. @Column(name="root\_channel\_id")
61. **private** String rootChannelId;



66. **public** Integer getIdBatch() {
67. **return** idBatch;
68. }
70. **public** **void** setIdBatch(Integer idBatch) {
71. **this**.idBatch = idBatch;
72. }
74. **public** String getChannelId() {
75. **return** channelId;
76. }
78. **public** **void** setChannelId(String channelId) {
79. **this**.channelId = channelId;
80. }
82. **public** Date getLogDate() {
83. **return** logDate;
84. }
86. **public** **void** setLogDate(Date logDate) {
87. **this**.logDate = logDate;
88. }
90. **public** String getLoggingObjectType() {
91. **return** loggingObjectType;
92. }
94. **public** **void** setLoggingObjectType(String loggingObjectType) {
95. **this**.loggingObjectType = loggingObjectType;
96. }
98. **public** String getObjectName() {
99. **return** objectName;
100. }
102. **public** **void** setObjectName(String objectName) {
103. **this**.objectName = objectName;
104. }
106. **public** String getObjectCopy() {
107. **return** objectCopy;
108. }
110. **public** **void** setObjectCopy(String objectCopy) {
111. **this**.objectCopy = objectCopy;
112. }
114. **public** String getRepositoryDirectory() {
115. **return** repositoryDirectory;
116. }
118. **public** **void** setRepositoryDirectory(String repositoryDirectory) {
119. **this**.repositoryDirectory = repositoryDirectory;
120. }
122. **public** String getFilename() {
123. **return** filename;
124. }
126. **public** **void** setFilename(String filename) {
127. **this**.filename = filename;
128. }
130. **public** String getObjectId() {
131. **return** objectId;
132. }
134. **public** **void** setObjectId(String objectId) {
135. **this**.objectId = objectId;
136. }
138. **public** String getObjectRevision() {
139. **return** objectRevision;
140. }
142. **public** **void** setObjectRevision(String objectRevision) {
143. **this**.objectRevision = objectRevision;
144. }
146. **public** String getParentChannelId() {
147. **return** parentChannelId;
148. }
150. **public** **void** setParentChannelId(String parentChannelId) {
151. **this**.parentChannelId = parentChannelId;
152. }
154. **public** String getRootChannelId() {
155. **return** rootChannelId;
156. }
158. **public** **void** setRootChannelId(String rootChannelId) {
159. **this**.rootChannelId = rootChannelId;
160. }


164. }

## 2、然后，在Repository中写存储过程调用：

## @Procedure

**[java]** [view plain](http://blog.csdn.net/paopaomm/article/details/51851919) [copy](http://blog.csdn.net/paopaomm/article/details/51851919)

1. **package** com.xxxx.core.main.etl.repository;
3. **import** java.util.Date;
4. **import** java.util.List;
5. **import** org.springframework.data.jpa.repository.JpaRepository;
6. **import** org.springframework.data.jpa.repository.query.Procedure;
7. **import** org.springframework.data.repository.query.Param;
9. **public** **interface** JobchanLogRepository **extends** JpaRepository<JobchanLog, Date>{
11. List<JobchanLog> findByLogDateGreaterThan(Date logDate);
12. @Procedure(name = "p\_bakuplog")
13. String pBakuplog(@Param("flag") String flag);
15. }

## 3、最后在服务层调用上述函数即可：

**[java]** [view plain](http://blog.csdn.net/paopaomm/article/details/51851919) [copy](http://blog.csdn.net/paopaomm/article/details/51851919)

1. **package** com.xxxx.core.main.etl.service;

4. **import** java.net.InetAddress;
5. **import** java.net.UnknownHostException;
6. **import** java.util.ArrayList;
7. **import** java.util.Calendar;
8. **import** java.util.Date;
9. **import** java.util.HashMap;
10. **import** java.util.List;
11. **import** java.util.Map;
13. **import** org.apache.log4j.Logger;
14. **import** org.springframework.beans.factory.annotation.Autowired;
15. **import** org.springframework.stereotype.Service;
17. /\*\*
18. \* Log处理类
19. \*
20. \* @author mengpaopao
21. \* @date 20160603
22. \*/
23. @Service
24. **public** **class** LogService {
26. @Autowired
27. JobchanLogRepository jobchanLogRepository;
29. Map<String , Object>  map  = **new** HashMap<String, Object>();
31. **private** **static** **final** Logger LOGGER = Logger.getLogger(LogService.**class**);

34. /\*\*
35. \*  @Description:备份并清理日志
36. \*  @date:2016年07月06日
37. \*  @author:mengpaopao
38. \*  @param
39. \*  @return
40. \*/
41. **public** String backupLog(){
43. String flag = jobchanLogRepository.pBakuplog("1");
44. **if**("0".equals(flag)){
45. LOGGER.info("备份完成！");
46. }**else** **if**("1".equals(flag)){
47. LOGGER.info("已经备份过了！");
48. }**else** **if**("2".equals(flag)){
49. LOGGER.info("备份异常！");
50. }
51. **return** flag;
52. }
54. }

其他略。

# 2、in out两种存储过程

<http://blog.csdn.net/chszs/article/details/50127823>

### 1、这里有两个存储过程：

CREATE OR REPLACE PACKAGE test\_pkg AS

PROCEDURE in\_only\_test (inParam1 IN VARCHAR2);

PROCEDURE in\_and\_out\_test (inParam1 IN VARCHAR2, outParam1 OUT VARCHAR2);

END test\_pkg;

/

CREATE OR REPLACE PACKAGE BODY test\_pkg AS

PROCEDURE in\_only\_test(inParam1 IN VARCHAR2) AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('in\_only\_test');

END in\_only\_test;

PROCEDURE in\_and\_out\_test(inParam1 IN VARCHAR2, outParam1 OUT VARCHAR2) AS

BEGIN

outParam1 := 'Woohoo Im an outparam, and this is my inparam ' || inParam1;

END in\_and\_out\_test;

END test\_pkg;

1）in\_only\_test   
它需要一个输入参数inParam1，但不返回值

2）in\_and\_out\_test   
它需要一个输入参数inParam1，且返回值outParam1

### 2、@NamedStoredProcedureQueries

我们可以使用@NamedStoredProcedureQueries注释来调用存储过程。

@Entity

@Table(name = "MYTABLE")

@NamedStoredProcedureQueries({

@NamedStoredProcedureQuery(name = "in\_only\_test", procedureName = "test\_pkg.in\_only\_test", parameters = {

@StoredProcedureParameter(mode = ParameterMode.IN, name = "inParam1", type = String.class) }),

@NamedStoredProcedureQuery(name = "in\_and\_out\_test", procedureName = "test\_pkg.in\_and\_out\_test", parameters = {

@StoredProcedureParameter(mode = ParameterMode.IN, name = "inParam1", type = String.class),

@StoredProcedureParameter(mode = ParameterMode.OUT, name = "outParam1", type = String.class) }) })

public class MyTable implements Serializable {

}

关键要点：

* 存储过程使用了注释@NamedStoredProcedureQuery，并绑定到一个JPA表。
* procedureName是存储过程的名字
* name是JPA中的存储过程的名字
* 使用注释@StoredProcedureParameter来定义存储过程使用的IN/OUT参数

### 3、创建Spring Data JPA数据库

下面我们来创建Spring Data JPA数据库：

public interface MyTableRepository extends CrudRepository<MyTable, Long> {

@Procedure(name = "in\_only\_test")

void inOnlyTest(@Param("inParam1") String inParam1);

这里的输出参数不需要传入，所以这里没有写上输出参数，但是上面的声明中已经具备了

@Procedure(name = "in\_and\_out\_test")

String inAndOutTest(@Param("inParam1") String inParam1);

}

### 关键要点：

* @Procedure的name参数必须匹配@NamedStoredProcedureQuery的name
* @Param必须匹配@StoredProcedureParameter注释的name参数
* 返回类型必须匹配：in\_only\_test存储过程返回是void，in\_and\_out\_test存储过程必须返回String