**[Spring JdbcTemplate使用别名传参（NamedParameterJdbcTemplate）](http://blog.csdn.net/victor_cindy1/article/details/52191128)**

标签： [NamedParameterJdbcTe](http://www.csdn.net/tag/NamedParameterJdbcTe)[JdbcTemplate](http://www.csdn.net/tag/JdbcTemplate)

2016-08-12 14:24 1332人阅读 [评论](http://blog.csdn.net/victor_cindy1/article/details/52191128#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/victor_cindy1/article/details/52191128#report)

 分类：

Spring（29）

版权声明：本文为博主原创文章，未经博主允许不得转载。

在使用JdbcTemplate时，一般传参都是用的？来绑定参数，但是对于某种情况就不适用了，例如Sql中如果存在IN，那么写SQL的时候就会比较麻烦，例如，咱们要查ID在某个范围内的数据，一般情况下咱们这么写：

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. List<String> ids = **new** ArrayList<String>();
2. ids.add("id1");
3. ids.add("id2");
4. ids.add("id3");
5. String sql = "SELECT \* FROM TEST WHERE ID IN(";
6. **for**(**int** i = 0; i<ids.size();i++){
7. sql+="?";
8. **if**(i!=ids.size()){
9. sql+=",";
10. }
11. }
12. sql+=")";
13. List result = getJdbcTemplate.queryForList(sql, ids.toArray());

这里需要提到的类就是NamedParameterJdbcTemplate，他是Spring给开发者提供的一个基于JdbcTemplate的类，他支持命名参数特性。包含了JdbcTemplate中的大部分方法，主要有三类：execute方法、query及queryForXXX方法、update及batchUpdate方法。  
  
咱们可以看一个由NamedParameterJdbcTemplate完成的上述例子：

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. List<String> ids = **new** ArrayList<String>();
2. ids.add("id1");
3. ids.add("id2");
4. ids.add("id3");
5. String sql = "SELECT \* FROM TEST WHERE ID IN(:ids)";
6. Map params = **new** HashMap();
7. params.addValue("ids", ids);
8. NamedParameterJdbcTemplate jdbcTemplate = **null**;
9. jdbcTemplate = **new** NamedParameterJdbcTemplate(getJdbcTemplate());
10. List reslut = jdbcTemplate.query(sql, params);

可以看到Sql中的参数可以用 :[name] 的方式书写，在执行之前，所有的参数可以放在一个Map中，key为Sql中的参数名，这样的话，就简化了咱们自己拼写Sql的工作量。  
其实，这只是NamedParameterJdbcTemplate的好处之一，另一个好处就是，对于同一个参数，多次出现在一条sql中时，也很好处理。  
例如咱们要查一个数据，传入的时间参数要在字段F1和字段F2之间。那么可以这么写：

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. String sql = "SELECT \* FROM TEST WHERE F1>:time and F2<:time";
2. Map params = **new** HashMap();
3. params.addValue("time", **new** Date());
4. NamedParameterJdbcTemplate jdbcTemplate = **null**;
5. jdbcTemplate = **new** NamedParameterJdbcTemplate(getJdbcTemplate());
6. List reslut = jdbcTemplate.query(sql, params);

上面咱们利用NamedParameterJdbcTemplate查询时，是将所有参数放入到了一个Map中，其实NamedParameterJdbcTemplate为咱们提供的参数模型不止Map。还有SqlParameterSource和BeanPropertySqlParameterSource。  
其中SqlParameterSource和咱们用Map一样，他只是对Map进行了封装。  
而BeanPropertySqlParameterSource封装了一个JavaBean对象，通过JavaBean对象属性来决定命名参数的值。  
例如咱们创建了一个Bean。

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. **public** **class** User {
2. **private** **int** id;
3. **private** String userName;
4. **private** String password;
5. //省略getter和setter
6. }

现在验证一个登录信息是否正确：

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. **public** **boolean** login(String userName,String password){
2. NamedParameterJdbcTemplate namedParameterJdbcTemplate = **null**;
3. namedParameterJdbcTemplate = **new** NamedParameterJdbcTemplate(getJdbcTemplate());
4. User user = **new** User();
5. model.setUserName("jialeens");
6. model.setPassword("hehe");
7. String sql = "SELECT COUNT(1) FROM USER WHERE USERNAME=:userName AND PASSWORD=:password";
8. SqlParameterSource paramSource = **new** BeanPropertySqlParameterSource(user);
9. **int** size = namedParameterJdbcTemplate.queryForInt(sql, paramSource);
10. **return** size==1;
11. }

可以看到，传入的参数是一个对象Bean，Sql在执行时，会根据Sql中的参数名去获取对应Bean中的属性。

NamedParameterJdbcTemplate内部包含了一个JdbcTemplate，所以JdbcTemplate能做的事情NamedParameterJdbcTemplate都能干； NamedParameterJdbcTemplate相对于JdbcTemplate主要增加了参数可以命名的功能。  
public Object queryForObject(String sql, Map paramMap, RowMapper rowMapper)  
public Object queryForObject(String sql, SqlParameterSource paramSource, RowMapper rowMapper)  
SqlParameterSource的两个主要实现MapSqlParameterSource  
和BeanPropertySqlParameterSource  
public int update(String sql, SqlParameterSource paramSource, KeyHolder generatedKeyHolder)保存数据获得主键。

**[java]** [view plain](http://blog.csdn.net/victor_cindy1/article/details/52191128) [copy](http://blog.csdn.net/victor_cindy1/article/details/52191128)

1. **public** **class** NamedJdbcTemplate {
2. // JdbcTemplate是线程安全的
3. **static** JdbcTemplate jdbc = **new** JdbcTemplate(JdbcUtils.getDataSource());
4. **static** NamedParameterJdbcTemplate named = **new** NamedParameterJdbcTemplate(
5. JdbcUtils.getDataSource());
6. **public** **static** **void** main(String[] args) {
7. User user=**new** User();
8. user.setMoney(10);
9. user.setId(2);
10. //  System.out.println(findUser(user));
11. System.out.println(findUser1(user));
12. }
13. **static** **void** addUser(User user){
14. String sql = "insert into user(name,birthday, money) values (:name,:birthday,:money) ";//：后的命名要与列名一致
15. SqlParameterSource ps=**new** BeanPropertySqlParameterSource(user);//从user中取出数据，与sql语句中一一对应将数据换进去
16. KeyHolder keyHolder=**new** GeneratedKeyHolder();
17. named.update(sql, ps, keyHolder);
18. **int** id=keyHolder.getKey().intValue();//获得主键
19. user.setId(id);
20. //Map map=keyholder.getKeys();//这样可以得到联合主键的值
21. //keyholder.getKeyList();//这样可以得到一些主主键值，若一次添加好几条记录
22. }
23. **static** User findUser1(User user) {
24. String sql = "select id, name, money, birthday  from user where money>:money and id<:id";
25. SqlParameterSource ps=**new** BeanPropertySqlParameterSource(user);
26. Object u=named.queryForObject(sql, ps, **new** BeanPropertyRowMapper(User.**class**));
27. **return** (User) u;
28. }
29. **static** User findUser(User user) {
30. String sql = "select id, name, money, birthday  from user where money>:m and id<:id";
31. Object[] args = **new** Object[] {user.getName(),user.getMoney(),user.getId() };
32. Map params=**new** HashMap();
33. params.put("m", user.getMoney());
34. params.put("id", user.getId());
35. Object u=named.queryForObject(sql, params, **new** BeanPropertyRowMapper(User.**class**));
36. **return** (User) u;
37. }