# SQLQuery对象

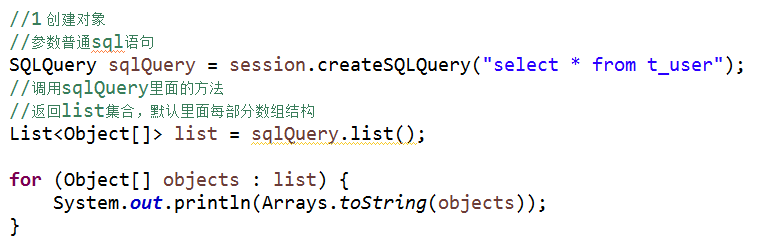
## 概述

底层sql实现

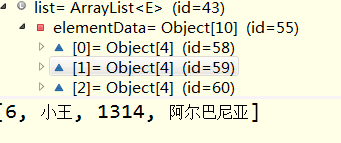
## 实现过程

（1）创建对象

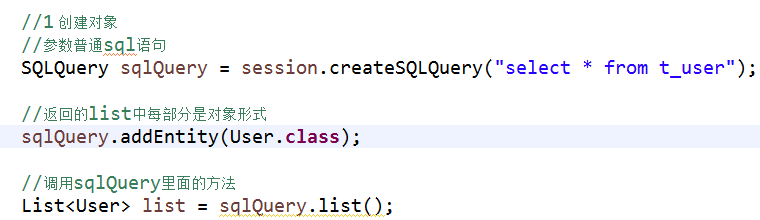
（2）调用对象的方法得到结果

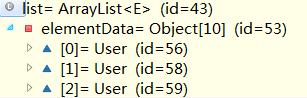


返回list集合每部分是数组



返回list中每部分是对象形式





## 方法

由于返回结果都是数组类型，就需要方法对结果进行转化

### 添加方法

.addScalar()

.addEntity

addJoin()

### 返回方法

.setResultTransformer(Transformers.aliasToBean(CatDTO.class))

# 查询返回--list

## 标量查询

最基本的SQL查询就是获得一个标量（数值）的列表。

sess.createSQLQuery("SELECT \* FROM CATS").list();

sess.createSQLQuery("SELECT ID, NAME, BIRTHDATE FROM CATS").list();

它们都将返回一个Object数组(Object[])组成的List，数组每个元素都是CATS表的一个字段值。Hibernate会使用ResultSetMetadata来判定返回的标量值的实际顺序和类型。

如果要避免过多的使用ResultSetMetadata,或者只是为了更加明确的指名返回值，可以使用addScalar()。

sess.createSQLQuery("SELECT \* FROM CATS").addScalar("ID", Hibernate.LONG).addScalar("NAME", Hibernate.STRING).addScalar("BIRTHDATE", Hibernate.DATE)

这个查询指定了:

* SQL查询字符串
* 要返回的字段和类型

它仍然会返回Object数组,但是此时不再使用ResultSetMetdata,而是明确的将ID,NAME和BIRTHDATE按照Long,String和Short类型从resultset中取出。同时，也指明了就算query是使用\*来查询的，可能获得超过列出的这三个字段，也仅仅会返回这三个字段。

对全部或者部分的标量值不设置类型信息也是可以的。

sess.createSQLQuery("SELECT \* FROM CATS") .addScalar("ID", Hibernate.LONG) .addScalar("NAME") .addScalar("BIRTHDATE")

基本上这和前面一个查询相同,只是此时使用ResultSetMetaData来决定NAME和BIRTHDATE的类型，而ID的类型是明确指出的。

关于从ResultSetMetaData返回的java.sql.Types是如何映射到Hibernate类型，是由方言(Dialect)控制的。假若某个指定的类型没有被映射，或者不是你所预期的类型，你可以通过Dialet的registerHibernateType调用自行定义。

## 实体查询

上面的查询都是返回标量值的，也就是从resultset中返回的“裸”数据。下面展示如何通过addEntity()让原生查询返回实体对象。

sess.createSQLQuery("SELECT \* FROM CATS").addEntity(Cat.class);

sess.createSQLQuery("SELECT ID, NAME, BIRTHDATE FROM CATS").addEntity(Cat.class);

这个查询指定：

* SQL查询字符串
* 要返回的实体

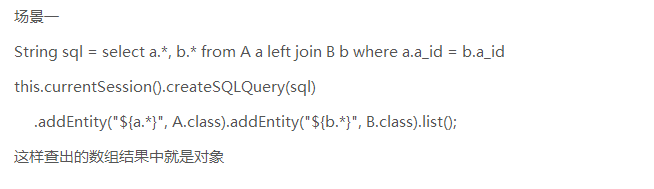
假设Cat被映射为拥有ID,NAME和BIRTHDATE三个字段的类，以上的两个查询都返回一个List，每个元素都是一个Cat实体。

假若实体在映射时有一个many-to-one的关联指向另外一个实体，在查询时必须也返回那个实体，否则会导致发生一个"column not found"的数据库错误。这些附加的字段可以使用\*标注来自动返回，但我们希望还是明确指明，看下面这个具有指向Dog的many-to-one的例子：

sess.createSQLQuery("SELECT ID, NAME, BIRTHDATE, DOG\_ID FROM CATS").addEntity(Cat.class);

这样cat.getDog()就能正常运作。

### 多表返回



## 处理关联和集合类—了解

通过提前抓取将Dog连接获得，而避免初始化proxy带来的额外开销也是可能的。这是通过addJoin()方法进行的，这个方法可以让你将关联或集合连接进来。

sess.createSQLQuery("SELECT c.ID, NAME, BIRTHDATE, DOG\_ID, D\_ID, D\_NAME FROM CATS c, DOGS d WHERE c.DOG\_ID = d.D\_ID").addEntity("cat", Cat.class).addJoin("cat.dog");

上面这个例子中，返回的Cat对象，其dog属性被完全初始化了，不再需要数据库的额外操作。注意，我们加了一个别名("cat")，以便指明join的目标属性路径。通过同样的提前连接也可以作用于集合类，例如，假若Cat有一个指向Dog的一对多关联。

sess.createSQLQuery("SELECT ID, NAME, BIRTHDATE, D\_ID, D\_NAME, CAT\_ID FROM CATS c, DOGS d WHERE c.ID = d.CAT\_ID") .addEntity("cat", Cat.class) .addJoin("cat.dogs");

到此为止，我们碰到了天花板：若不对SQL查询进行增强，这些已经是在Hibernate中使用原生SQL查询所能做到的最大可能了。

下面的问题即将出现：返回多个同样类型的实体怎么办？或者默认的别名/字段不够又怎么办？

## 返回多个实体

到目前为止,结果集字段名被假定为和映射文件中指定的的字段名是一致的。假若SQL查询连接了多个表，同一个字段名可能在多个表中出现多次，这就会造成问题。

下面的查询中需要使用字段别名注射（这个例子本身会失败）：

sess.createSQLQuery("SELECT c.\*, m.\* FROM CATS c, CATS m WHERE c.MOTHER\_ID = c.ID") .addEntity("cat", Cat.class) .addEntity("mother", Cat.class)

这个查询的本意是希望每行返回两个Cat实例，一个是cat,另一个是它的妈妈。但是因为它们的字段名被映射为相同的，而且在某些数据库中，返回的字段别名是“c.ID”,"c.NAME"这样的形式，而它们和在映射文件中的名字（"ID"和"NAME"）不匹配，这就会造成失败。

下面的形式可以解决字段名重复：

sess.createSQLQuery("SELECT {cat.\*}, {mother.\*} FROM CATS c, CATS m WHERE c.MOTHER\_ID = c.ID") .addEntity("cat", Cat.class) .addEntity("mother", Cat.class)

这个查询指明：

* SQL查询语句，其中包含占位附来让Hibernate注射字段别名
* 查询返回的实体

上面使用的{cat.\*}和{mother.\*}标记是作为“所有属性”的简写形式出现的。当然你也可以明确地罗列出字段名，但在这个例子里面我们让Hibernate来为每个属性注射SQL字段别名。字段别名的占位符是属性名加上表别名的前缀。在下面的例子中，我们从另外一个表（cat\_log）中通过映射元数据中的指定获取Cat和它的妈妈。注意，要是我们愿意，我们甚至可以在where子句中使用属性别名。

String sql = "SELECT ID as {c.id}, NAME as {c.name}, " + "BIRTHDATE as {c.birthDate}, MOTHER\_ID as {c.mother}, {mother.\*} " + "FROM CAT\_LOG c, CAT\_LOG m WHERE {c.mother} = c.ID"; List loggedCats = sess.createSQLQuery(sql) .addEntity("cat", Cat.class) .addEntity("mother", Cat.class).list()

### 别名和属性引用(Alias and property references)

大多数情况下，都需要上面的属性注射，但在使用更加复杂的映射，比如复合属性、通过标识符构造继承树，以及集合类等等情况下，也有一些特别的别名，来允许Hibernate注射合适的别名。

下表列出了使用别名注射参数的不同可能性。注意：下面结果中的别名只是示例，实用时每个别名需要唯一并且不同的名字。

表 16.1. 别名注射(alias injection names)

| **描述** | **语法** | **示例** |
| --- | --- | --- |
| 简单属性 | {[aliasname].[propertyname] | A\_NAME as {item.name} |
| 复合属性 | {[aliasname].[componentname].[propertyname]} | CURRENCY as {item.amount.currency}, VALUE as {item.amount.value} |
| 实体辨别器(Discriminator of an entity) | {[aliasname].class} | DISC as {item.class} |
| 实体的所有属性 | {[aliasname].\*} | {item.\*} |
| 集合键(collection key) | {[aliasname].key} | ORGID as {coll.key} |
| 集合id | {[aliasname].id} | EMPID as {coll.id} |
| 集合元素 | {[aliasname].element} | XID as {coll.element} |  |
| 集合元素的属性 | {[aliasname].element.[propertyname]} | NAME as {coll.element.name} |  |
| 集合元素的所有属性 | {[aliasname].element.\*} | {coll.element.\*} |  |
| 集合的所有属性 | {[aliasname].\*} | {coll.\*} |  |

## 返回非受管实体

可以对原生sql 查询使用ResultTransformer。这会返回不受Hibernate管理的实体。

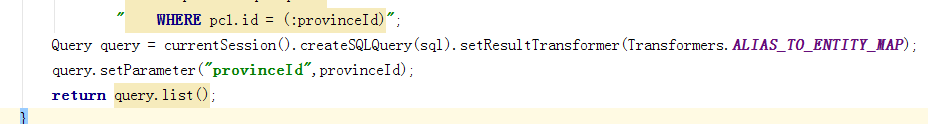
1http://www.blogjava.net/Images/OutliningIndicators/None.gifsess.createSQLQuery("SELECT NAME, BIRTHDATE FROM CATS")  
2http://www.blogjava.net/Images/OutliningIndicators/None.gif         .setResultTransformer(Transformers.aliasToBean(CatDTO.class))

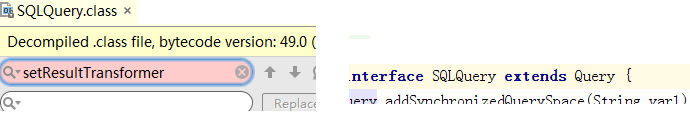
这个查询指定：SQL查询字符串,结果转换器(result transformer)

上面的查询将会返回CatDTO的列表,它将被实例化并且将NAME和BIRTHDAY的值注射入对应的属性或者字段。

# 奇葩--返回list<map>

## 父类才有这个方法接受





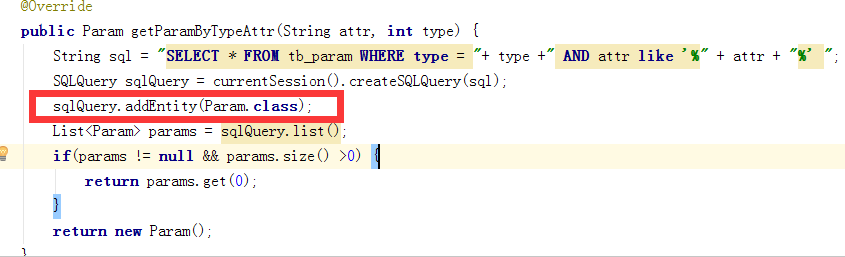
# 增删改

用exexcuteUpdate()接受就行，强转为Long便可

# 单表查询

参看：https://blog.csdn.net/xwnxwn/article/details/60333637

## 查询所有



## 投影查询，部分查询

