Hibernate 5 - Native SQL query example

Hibernate allows us to execute the native SQL queries for all create, update, delete and retrieve operations. This is useful if you want to improve the performance of your application using database specific queries.

In hibernate, you can execute your native SQL queries using the Session.createNativeQuery() method.

In this post, you will learn how to -

* Execute simple and complex native SQL query in hibernate application.
* Map native SQL query result to an @Entity object.
* Use parameter binding in native SQL queries.

Tools and technologies used for this application are -

* Hibernate ORM 5.2.17.Final
* JAXB API 2.3.0
* JavaSE 9
* MySQL Server 5.7.12
* Eclipse Oxygen.1
* Maven 3.3.9

Jar dependencies

Add the following jar dependencies in pom.xml file of your maven project.

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.17.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>6.0.6</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.0</version>

</dependency>

</dependencies>

Database table creation and Entity class

Before we start working on native SQL queries, we need to create the database tables and @Entity classes for our examples.

So first execute the following DDL and DML statements in your MySQL server.

**create** **table** DEPARTMENT(

DPT\_ID **int** **not** **null** **primary** **key**,

NAME **varchar**(100) **not** **null**

);

**create** **table** EMPLOYEE(

EMP\_ID **int** **not** **null** **primary** **key**,

NAME **varchar**(100) **not** **null**,

DESIGNATION **varchar**(100),

DPT\_ID **int**,

**foreign** **key** (DPT\_ID) **references** DEPARTMENT(DPT\_ID)

);

**insert** **into** DEPARTMENT(DPT\_ID,NAME) **values**(1,'Software Development');

**insert** **into** DEPARTMENT(DPT\_ID,NAME) **values**(2,'Human Resources');

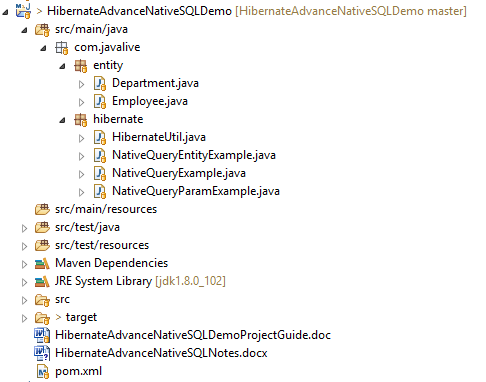
**insert** **into** EMPLOYEE(EMP\_ID,NAME,DESIGNATION,DPT\_ID) **values**(1,'Mike','Software Developer',1);

**insert** **into** EMPLOYEE(EMP\_ID,NAME,DESIGNATION,DPT\_ID) **values**(2,'David','Team Lead',1);

**insert** **into** EMPLOYEE(EMP\_ID,NAME,DESIGNATION,DPT\_ID) **values**(3,'Peter','Manager',2);

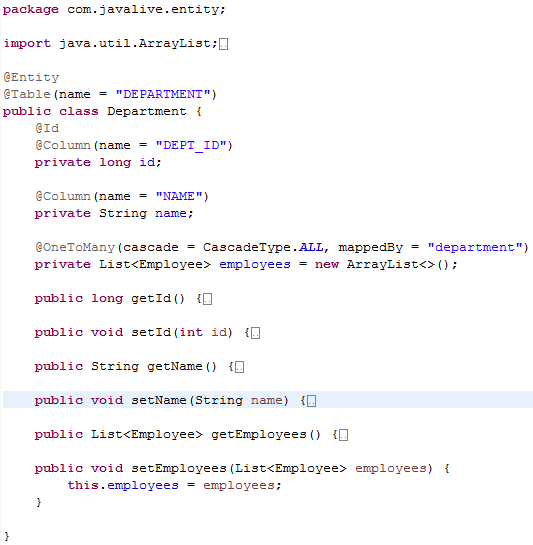
**insert** **into** EMPLOYEE(EMP\_ID,NAME,DESIGNATION,DPT\_ID) **values**(4,'Andrew','VP',2);

Project Structure :-

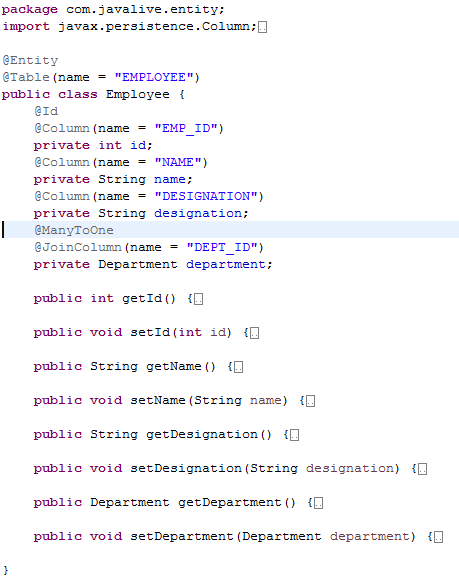


Now, create Department and Employee entities to map with DEPARTMENT and EMPLOYEE tables respectively.

**Department.java**

****

**Employee.java**

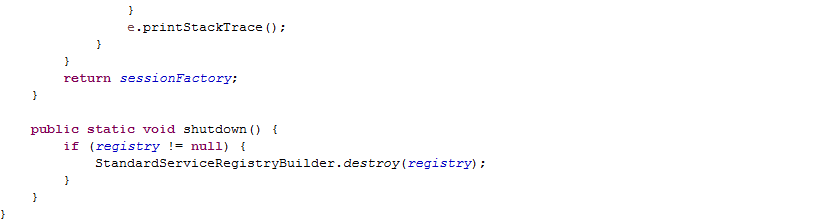
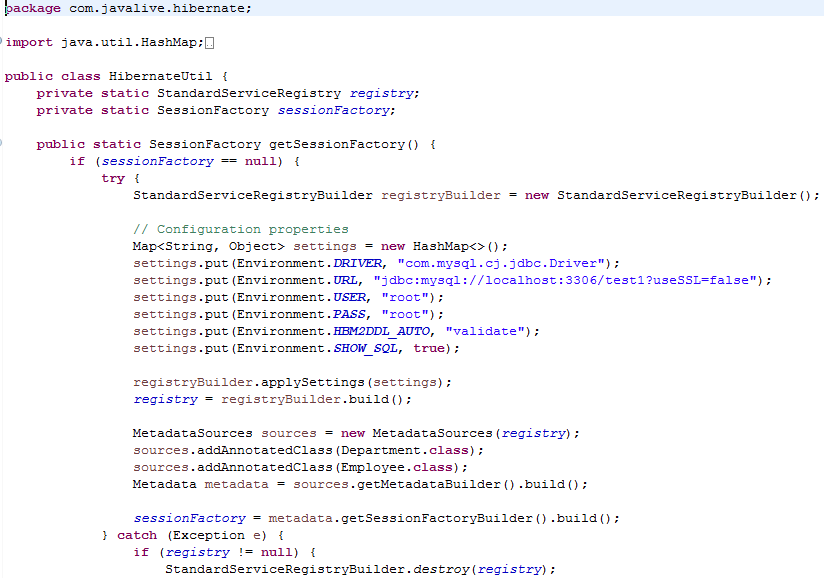


Hibernate Utility class

Create a helper class HibernateUtil to bootstrap hibernate application.

Map the Department and Employee entities using the #MetadataSources.addAnnotatedClass() method.

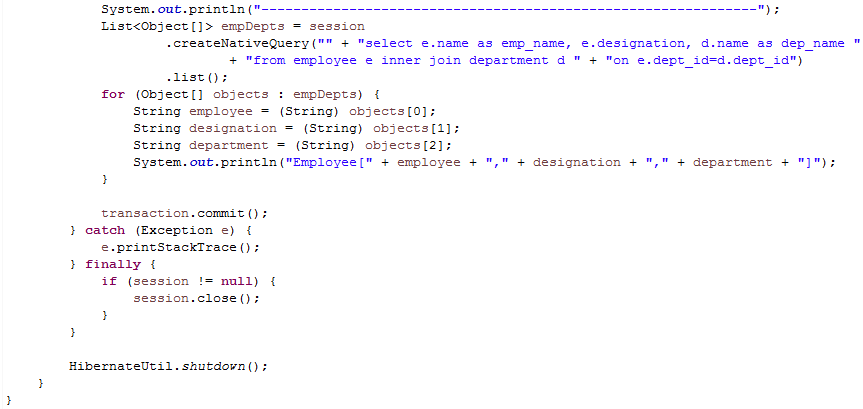
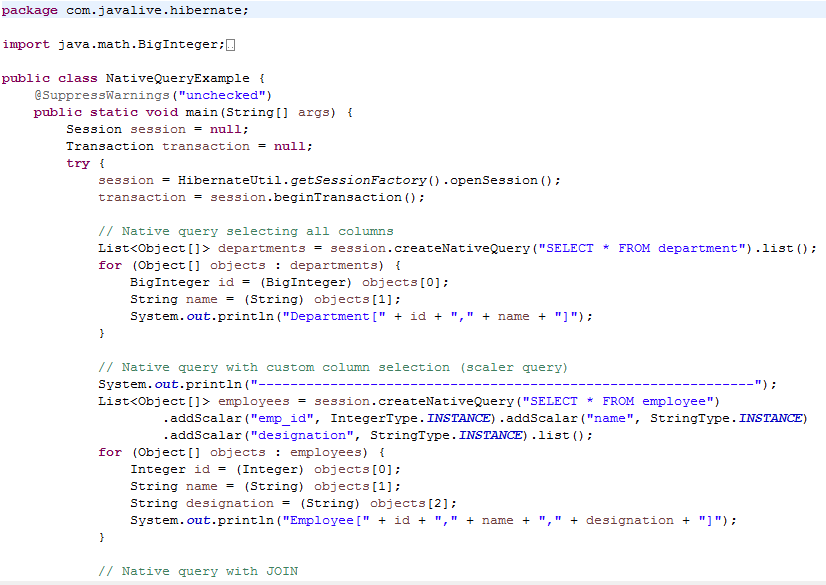
**HibernateUtil.java**



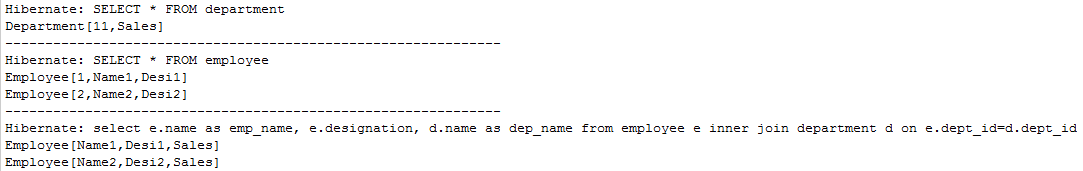
Using native SQL query in Hibernate

The following example demonstrates how to use the session.createNativeQuery() method to execute the simple and complex queries in hibernate.

**NativeQueryExample.java**



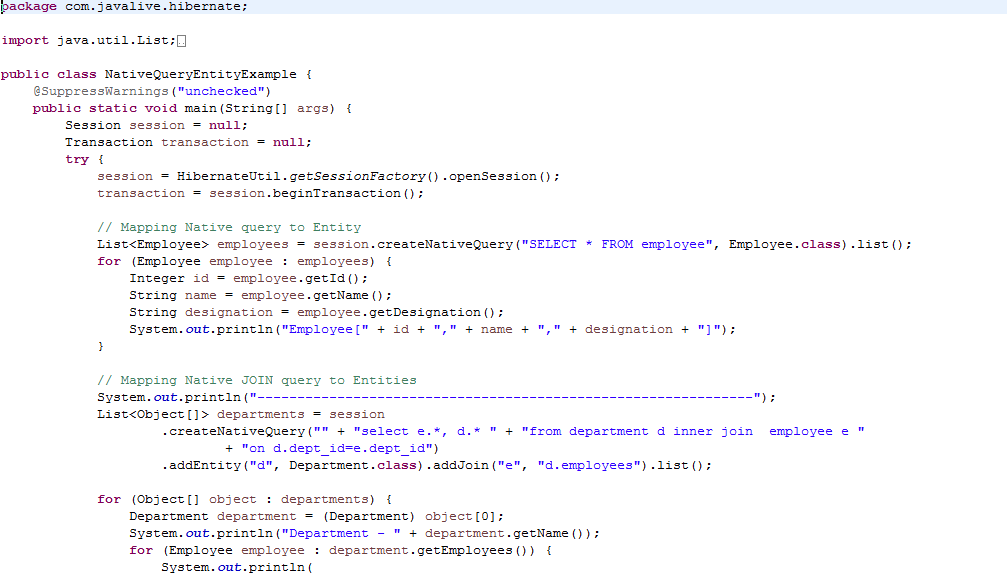
**Output**

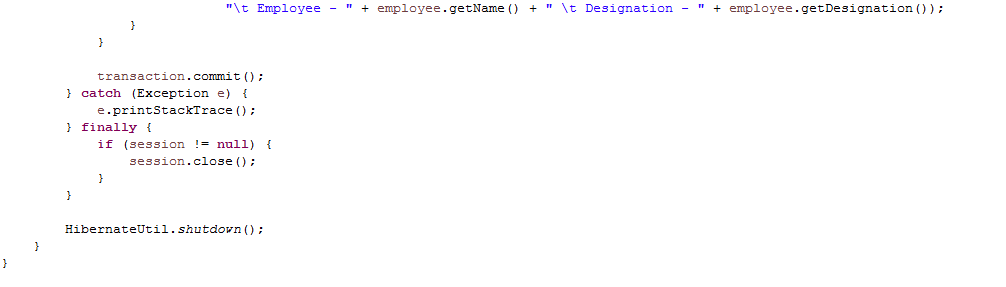


Mapping native SQL query result to Entity object

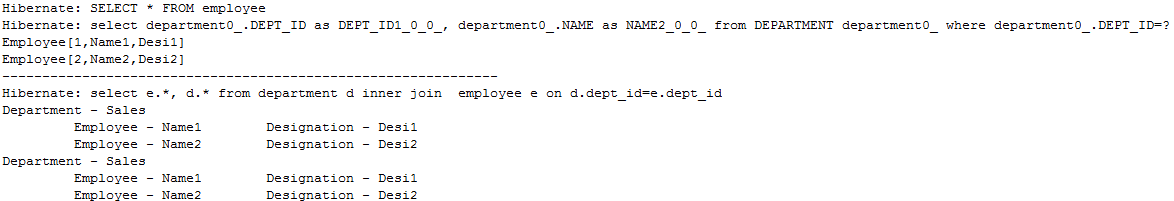
The following example demonstrates how to map the native SQL query result to an entity object by specifying the second parameter in  the session.createNativeQuery() method or using the NativeQuery.addEntity() method.

**NativeQueryEntityExample.java**





**Output**



Parameter binding with native SQL query

This example demonstrates how to use the positional and named parameter binding with native SQL queries.

**NativeQueryParamExample.java**



**Output**

