Hibernate Querying

Agenda

1. Hibernate Querying
2. Hibernate Query Language (HQL)
3. CriteriaQuery
4. Native SQL

Hibernate Querying

Introduction

- Session objects **get** and **load** method helps us to query the database and retrieve data
- But they can retrieve exactly one object based on the id column
- For retrieving multiple records with different restrictions and conditions Hibernate provides different Interfaces
- From Hibernate 5.2 these are Typed Interfaces

1. Hibernate Query Language(HQL) interface Query
2. CriteriaQuery
3. NativeSQL

Hibernate Query Language (HQL)

Introduction

- HQL is a powerful query language used to execute queries against database
- HQL is much like SQL, representing it in the form of Objects (Java Persistent Class Object)
- HQL uses the names of Java classes and properties instead of tables and columns
- And It is case-insensitive
- It returns result as an object or collection; So we don't have to copy data from ResultSet to bean objects
- HQL Queries are Database Independent
- It's as simple as this to print all the records from Student_tbl table

```
org.hibernate.query.Query<Student> query = session.createQuery("From Student");
List<Student> list = query.list();
System.out.println(list);
```

Persistent Class Name

Hibernate Query Language (HQL)

Advance Features

- org.hibernate.Query is deprecated with Hibernate 5.2
- In our examples we'll be using org.hibernate.query.Query<R>
- HQL also supports advanced features like
  - Pagination
  - joins with dynamic profiling (Inner/outer/full joins/ Cartesian products)
  - Projection
  - Aggregation and grouping
  - Ordering
  - Sub queries and
  - SQL function calls

Hibernate Query Language (HQL)

Some frequently used Clauses in HQL

- FROM
- SELECT
- WHERE
- ORDER BY
- GROUP BY
- These Keywords are *case-insensitive*
- where as Class names and variables are *case-sensitive*

For the HQL Query Demo Guide Refer : Hibernate Query Language.pdf

Hibernate Query Language (HQL)

Legacy Parameter Binding

- HQL allows you to bind parameters in the query
- different set of values can be given to the same query
- 2 types of parameter binding are supported in HQL
- But these are deprecated since Hibernate 5.2
  - Named Parameter Binding

```
Query<Student> query = session.createQuery("From Student as std where std.studentName=:name");
query.setString(“name”,”Harry”);
List<Student> list = query.list();
System.out.println(list);
```

  - JDBC Parameter Binding

```
Query query = session.createQuery("From Student as std where std.studentName= ?");
query.setString(0,”Harry”);
List<Student> list = query.list();
System.out.println(list);
```

Hibernate Query Language (HQL)

Parameter Binding

- Hibernate 5.2 no longer supports named parameter binding or legacy JDBC binding
- It uses JPA-style ordinal parameters like

  “From Student std where std.course = ?0 and std.studentName = ?1”
- To bind the values to the ‘?’ positions setParameter method is used.

  setParameter(int position, java.lang.Object val)

```
Query<Student> query = session.createQuery("From Student std where std.course = ?0");
query.setParameter(0,course);
List<Student> list = query.list();
System.out.println(list);
```

```
Hibernate: select student0_.stdid as stdid1_0_, student0_.course as course2_0_, student0_.stdName as stdName3_0_
        from STUDENT_TBL student0_ where student0_.course=?
JAVA Student Records
Student [studentId=101, studentName=Harry, course=Java]
Student [studentId=106, studentName=ALLEN, course=Java]
**************************************************
```

Hibernate Query Language (HQL)

Few HQL methods

- setFirstResult – sets the starting row
- setMaxResults – sets the maximum records to be retrieved

```
Query<Student> query = session.createQuery("from Student");
query.setFirstResult(4);   // Starting row
query.setMaxResults(3);    // Size of each page
List<Student> list = query.list();
```

Criteria Queries

Introduction

- CriteriaQuery API helps you to build nested, structured query expressions in Java
- Provides compile-time syntax-checking which is not possible with HQL or SQL
- The legacy Criteria example given here is deprecated with Hibernate 5.2

```
Criteria crit = session.createCriteria(Student.class);
crit.add(Restrictions.like("studentName", "H%"));
List<Student> =  crit.list();
```

- CriteriaQuery has taken a complete new form with Hibernate 5.2

Criteria Queries

Simple Criteria Demo to Select all the records

- Like legacy Criteria we cannot directly create Criteria Query object
- To build a CriteriaQuery we need to first get the CriteriaBuilder from the session
- CriteriaBuilder provides the conditions and restrictions to the query

javax.persistence.criteria.CriteriaBuilder
javax.persistence.criteria.CriteriaQuery

```
CriteriaBuilder cb = session.getCriteriaBuilder();
CriteriaQuery<Student> cr = cb.createQuery(Student.class);
Root<Student> root = cr.from(Student.class);
cr.select(root);

Query<Student> query = session.createQuery(cr);
List<Student> list = query.list();
```

Criteria Queries

Why Criteria Over HQL?

- Criteria queries express the Query itself by
  - Programmatic
  - Type-safe way
- All conditions like where clause, order by, joins etc. need not be given in SQL query form
- All SQL query structure clauses are represented through classes and interfaces
- Criteria in Hibernate uses the JPA API over hibernate Criteria API for neat representation like
  - CriteriaBuilder
  - CriteriaQuery
  - Root
  - Join
  - Path
  - Predicate

Criteria Queries

Simple Criteria Demo to Select records with Condition

- conditions are expressed using CriteriaBuilder which returns a *predicate* object
- This *predicate* object is add to the Query by using *where* method of *CriteriaQuery*
- ***Root*** is used to define the base from which all attributes are taken
  - It also forms the basis for joins and paths

```
CriteriaBuilder cb = session.getCriteriaBuilder();
CriteriaQuery<Student> cr = cb.createQuery(Student.class);
Root<Student> root = cr.from(Student.class);
cr.select(root);
Predicate predicate = cb.equal(root.get("course"), “Oracle”);
cr.where(predicate);
Query<Student> query = session.createQuery(cr);
List<Student> result = query.list()
                    System.out.println("Student Records");
                    for(Student ob : result)
                            System.out.println(ob);
                    System.out.println("********************************************************");
```

For Detailed Demo On Criteria Refer:
Hibernate Criteria Query.pdf

Native SQL

Introduction

- Native SQL helps in creating typical Database SQL query from application
- Let us take a scenario, there is a Database specific feature which can be implemented as database query alone, to use its full feature in our application, we need to execute the Native SQL query
- Here in this case HQL or Criteria doesn't support
- You can create a native SQL query by using ***NativeQuery*** interface instantiated from the session object using *createNativeQuery()* method

```
String qry = "Select * from Student_TBL";
NativeQuery<Student> query = session.createNativeQuery(qry, Student.class);
List<Student> list = query.list();
```

Table Name

Type of return object instances

Summary

At the end of this module you were able to

- Query with Hibernate Query Language (HQL)
- Use Criteria Queries
- Create queries with Native SQL

Thank you