Hibernate Query Language

INTRODUCTION

Hibernate Query Language

OBJECTIVES

Using HQL in Hibernate applications.

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Define HQL
- Identify Advantages of using HQL
- Discuss various features of HQL
- Learn HQL Syntax for various database query operations
- Work with Named Queries
- Learn Associations and Joins

What is HQL? (Hibernate Query Language)

- Hibernate Query Language (HQL) is an object-oriented query language.
- Unlike native SQL query language which operate on tables and columns, HQL works with persistent objects and their properties.
- HQL queries are translated by Hibernate into conventional SQL queries which in turns perform action on database
- Keywords like SELECT , FROM and WHERE etc. are not case sensitive in HQL.
- Properties like table and column names are case sensitive.

Advantages of using HQL over traditional SQL

- Database Independent
- Easy to Learn for Java programmers

HQL From Clause

- Is used to load a complete persistent object into memory.

- From Clause Syntax :

```
String hql = "FROM Trainee";              // Trainee is the name of the Entity
Query query = session.createQuery(hql);
List results = query.list();
```

HQL As Clause

- Can be used to assign aliases to the classes in HQL queries.

```
String hql = "FROM Trainee AS t ";          // Trainee is the name of the Entity
Query query = session.createQuery(hql);
List results = query.list();
```

- The **AS** keyword is optional and you can also specify the alias directly after the class name, as follows:

```
String hql = "FROM Trainee  t ";
```

HQL SELECT Clause

- Provides more control over the resultset than the from clause.
- If you want to obtain few properties of the object instead of the complete object , use the SELECT clause.

```
String hql = "SELECT T.firstName FROM Trainee T";
Query query = session.createQuery(hql);
List results = query.list();
```

- In the above example it should be noted that T.firstName is the property of the Trainee object and not the firstName field of the Trainee Table.

HQL WHERE Clause

- Used to narrow retrieval of specific objects that are returned from the database.

```
String hql = "FROM Trainee T WHERE T.id = 30";
Query query = session.createQuery(hql);
List results = query.list();
```

HQL ORDER BY Clause

- Used to sort HQL query results .
- Results can be ordered by any property on the objects in the resultset either in ascending or descending order.
- Order By Syntax :

```
String hql = "FROM Trainee T WHERE t.id > 0 ORDER BY T.marks DESC";
Query query = session.createQuery(hql);
List results = query.list();
```

HQL GROUP BY Clause

- Used to pull information from database and group them based on a value of an attribute and typically use the result to include an aggregate value.

- Group By Syntax :

```
String hql = "SELECT SUM(T.marks), T.firtName FROM Trainee T  GROUP
BY  T.firstName";
Query query = session.createQuery(hql);
List results = query.list();
```

HQL Aggregate Functions

- HQL supports a range of aggregate methods, similar to SQL.
- They work the same way in HQL as in SQL .

Sr No	Function	Description
1	Avg(property name)	The Average of a property's value
2	Count(property name or *)	The number of times a property occurs in result
3	Max(property name)	The maximum value of the property values
4	Min(property name)	The minimum value of the property values
5	Sum (property name)	The sum of total property values

Example :

```
String query = "SELECT count(distinct T.firstName) FROM Trainee T;
```

Executing Queries using HQL

```
//Use of select in HQL
String SQL_QUERY ="select c.name from Course c";
Query query = s.createQuery(SQL_QUERY);
for(Iterator it=query.iterate();it.hasNext();)
  {
    String x =(String) it.next();
    System.out.println("Name: " + x);
    } tx.commit();
```

```
//Deleting a record using HQL
Query query=s.createQuery("delete from Course where id=1");
query.executeUpdate();
tx.commit();
System.out.println("Record Deleted Successfully");
```

```
//Use of where in HQL
Scanner scn = new Scanner(System.in);
System.out.println("Enter Course Id");
int x = scn.nextInt();
String SQL_QUERY ="from Course course where id="+x;
Query query = s.createQuery(SQL_QUERY);
for(Iterator it=query.iterate();it.hasNext();)
{
    Course course=(Course)it.next();
    System.out.println("Course ID: "+course.getId());
    System.out.println("Course Name: " + course.getName());
    System.out.println("Duration "+course.getDuration());

    }
tx.commit();
```

Named Queries using HQL

- Hibernate supports named parameters in its HQL queries.
- Accepting the user input from console is made easy by writing HQL queries.
- Developers does not have to defend against SQL injection attacks.

```
String hql = "FROM Employee E WHERE E.id = :employee_id";
Query query = session.createQuery(hql);
query.setParameter("employee_id",10);
List results = query.list();
```

Association using HQL

- HQL Association are used to retrieve the data from the multiple tables.
- Entities used can be associated using OnetoMany, ManytoOne, ManytoMany etc.,
- HQL joins are derived from ANSI SQL joins

Joins using HQL

- We can extract the details by joining two tables.
- HQL supports different types of join namely inner join and outer join
- Inner join :

Below Query shows how to inner join two table and get data from it.

```
Query query = session.createQuery("from Mother as m inner join m.childs "
        + "as c where m.motherId=4 and c.mother.motherId=4");
```

- Outer Join :

Below Query shows how to outer join two table and get data from it.

```
List<Object[]> allContinentsAndCountries = session.createQuery(
        "select cont.name, nvl(ctry.name, '[none]') " +
        "from Continent cont left join cont.countries ctry " +
        "with ctry.area > 100000 " +
        "order by cont.name")
        .list();
```

QUIZ QUESTION

Which of the following is correct HQL syntax for retrieving the Product entities from database ?

- ❑ SELECT * FROM product;
- ❑ SELECT ALL FROM Product;
- ❑ FROM Product
- ❑ FROM P AS Product;

SUMMARY

Hibernate Query Language

SUMMARY

In this lesson, you've learned to:

- Introduction to HQL (Hibernate Query Language)
- Advantage of Using HQL in Applications.
- FROM clause
- SELECT clause
- WHERE clause
- ORDER BY clause
- GROUP By clause
- HQL Aggregate Functions