**HQL - Hibernate Query Language - Example Tutorial**

HQL or Hibernate Query Language is the object-oriented query language of Hibernate Framework. HQL is very similar to SQL except that we use Objects instead of table names, that makes it more close to object oriented programming.

**Hibernate Query Language - HQL**

**HQL and Case Sensitivity**: HQL is case-insensitive except for java class and variable names. So SeLeCT is the same as sELEct is the same as SELECT, but com.journaldev.model.Employee is not same as com.journaldev.model.EMPLOYEE. Some of the commonly supported clauses in HQL are:

1. **HQL From**: HQL From is same as select clause in SQL, from Employee is same as select \* from Employee. We can also create alias such as from Employee emp or from Employee as emp.
2. **HQL Join** : HQL supports inner join, left outer join, right outer join and full join. For example, select e.name, a.city from Employee e INNER JOIN e.address a. In this query, Employee class should have a variable named address. We will look into it in the example code.
3. **Aggregate Functions**: HQL supports commonly used aggregate functions such as count(\*), count(distinct x), min(), max(), avg() and sum().
4. **Expressions**: HQL supports arithmetic expressions (+, -, \*, /), binary comparison operators (=, >=, <=, <>, !=, like), logical operations (and, or, not) etc.
5. HQL also supports ordre by and group by clauses.
6. HQL also supports sub-queries just like SQL queries.
7. HQL supports DDL, DML and executing store procedures too.

Let’s look at a simple example of using HQL in our program.

**HQL Example Database Setup**

I am using MySQL database for my example, below script will create two tables Employee and Address. They have one-to-one mapping and I am inserting some demo data for my example.

CREATE TABLE `Employee` (

`emp\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`emp\_name` varchar(20) NOT NULL,

`emp\_salary` double(10,0) NOT NULL DEFAULT '0',

PRIMARY KEY (`emp\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `Address` (

`emp\_id` int(11) unsigned NOT NULL,

`address\_line1` varchar(50) NOT NULL DEFAULT '',

`zipcode` varchar(10) DEFAULT NULL,

`city` varchar(20) DEFAULT NULL,

PRIMARY KEY (`emp\_id`),

CONSTRAINT `emp\_fk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `Employee` (`emp\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Employee` (`emp\_id`, `emp\_name`, `emp\_salary`)

VALUES

(1, 'Pankaj', 100),

(2, 'David', 200),

(3, 'Lisa', 300),

(4, 'Jack', 400);

INSERT INTO `Address` (`emp\_id`, `address\_line1`, `zipcode`, `city`)

VALUES

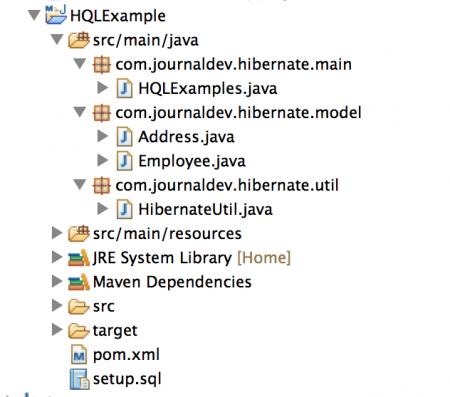
(1, 'Albany Dr', '95129', 'San Jose'),

(2, 'Arques Ave', '95051', 'Santa Clara'),

(3, 'BTM 1st Stage', '560100', 'Bangalore'),

(4, 'City Centre', '100100', 'New Delhi');

commit;

Create a maven project in Eclipse or the IDE you are using, our final project will look like below image.[](https://journaldev.nyc3.digitaloceanspaces.com/2014/05/HQL-Example-Project.png)

**Hibernate Maven Dependencies**

Our final pom.xml contains dependencies for Hibernate and MySQL driver.

<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.journaldev.hibernate</groupId>

<artifactId>HQLExample</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.5.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.0.5</version>

</dependency>

</dependencies>

</project>

**Hibernate Configuration XML**

Our hibernate configuration xml file contains database connection related properties and mapping classes. I will be using annotations for Hibernate mapping. hibernate.cfg.xml code:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"https://hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.password">pankaj123</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>

<property name="hibernate.connection.username">pankaj</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.current\_session\_context\_class">thread</property>

<property name="hibernate.show\_sql">true</property>

<mapping class="com.journaldev.hibernate.model.Employee"/>

<mapping class="com.journaldev.hibernate.model.Address"/>

</session-factory>

</hibernate-configuration>

**Hibernate SessionFactory Utility class**

We have a utility class to configure hibernate SessionFactory.

package com.journaldev.hibernate.util;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {

private static SessionFactory sessionFactory;

private static SessionFactory buildSessionFactory() {

try {

// Create the SessionFactory from hibernate.cfg.xml

Configuration configuration = new Configuration();

configuration.configure("hibernate.cfg.xml");

System.out.println("Hibernate Configuration loaded");

ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

System.out.println("Hibernate serviceRegistry created");

SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);

return sessionFactory;

}

catch (Throwable ex) {

System.err.println("Initial SessionFactory creation failed." + ex);

ex.printStackTrace();

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

if(sessionFactory == null) sessionFactory = buildSessionFactory();

return sessionFactory;

}

}

**Model Classes with Annotation based mapping**

Our model classes with JPA annotations looks like below.

package com.journaldev.hibernate.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import org.hibernate.annotations.Cascade;

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "emp\_id")

private long id;

@Column(name = "emp\_name")

private String name;

@Column(name = "emp\_salary")

private double salary;

@OneToOne(mappedBy = "employee")

@Cascade(value = org.hibernate.annotations.CascadeType.ALL)

private Address address;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

package com.journaldev.hibernate.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

import org.hibernate.annotations.GenericGenerator;

import org.hibernate.annotations.Parameter;

@Entity

@Table(name = "ADDRESS")

public class Address {

@Id

@Column(name = "emp\_id", unique = true, nullable = false)

@GeneratedValue(generator = "gen")

@GenericGenerator(name = "gen", strategy = "foreign",

parameters = { @Parameter(name = "property", value = "employee") })

private long id;

@Column(name = "address\_line1")

private String addressLine1;

@Column(name = "zipcode")

private String zipcode;

@Column(name = "city")

private String city;

@OneToOne

@PrimaryKeyJoinColumn

private Employee employee;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getAddressLine1() {

return addressLine1;

}

public void setAddressLine1(String addressLine1) {

this.addressLine1 = addressLine1;

}

public String getZipcode() {

return zipcode;

}

public void setZipcode(String zipcode) {

this.zipcode = zipcode;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public Employee getEmployee() {

return employee;

}

public void setEmployee(Employee employee) {

this.employee = employee;

}

}

**HQL Example Test Class**

Let’s see how to use HQL in java programs.

package com.journaldev.hibernate.main;

import java.util.Arrays;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import com.journaldev.hibernate.model.Employee;

import com.journaldev.hibernate.util.HibernateUtil;

public class HQLExamples {

@SuppressWarnings("unchecked")

public static void main(String[] args) {

//Prep work

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

Session session = sessionFactory.getCurrentSession();

//HQL example - Get All Employees

Transaction tx = session.beginTransaction();

Query query = session.createQuery("from Employee");

List<Employee> empList = query.list();

for(Employee emp : empList){

System.out.println("List of Employees::"+emp.getId()+","+emp.getAddress().getCity());

}

//HQL example - Get Employee with id

query = session.createQuery("from Employee where id= :id");

query.setLong("id", 3);

Employee emp = (Employee) query.uniqueResult();

System.out.println("Employee Name="+emp.getName()+", City="+emp.getAddress().getCity());

//HQL pagination example

query = session.createQuery("from Employee");

query.setFirstResult(0); //starts with 0

query.setFetchSize(2);

empList = query.list();

for(Employee emp4 : empList){

System.out.println("Paginated Employees::"+emp4.getId()+","+emp4.getAddress().getCity());

}

//HQL Update Employee

query = session.createQuery("update Employee set name= :name where id= :id");

query.setParameter("name", "Pankaj Kumar");

query.setLong("id", 1);

int result = query.executeUpdate();

System.out.println("Employee Update Status="+result);

//HQL Delete Employee, we need to take care of foreign key constraints too

query = session.createQuery("delete from Address where id= :id");

query.setLong("id", 4);

result = query.executeUpdate();

System.out.println("Address Delete Status="+result);

query = session.createQuery("delete from Employee where id= :id");

query.setLong("id", 4);

result = query.executeUpdate();

System.out.println("Employee Delete Status="+result);

//HQL Aggregate function examples

query = session.createQuery("select sum(salary) from Employee");

double sumSalary = (Double) query.uniqueResult();

System.out.println("Sum of all Salaries= "+sumSalary);

//HQL join examples

query = session.createQuery("select e.name, a.city from Employee e "

+ "INNER JOIN e.address a");

List<Object[]> list = query.list();

for(Object[] arr : list){

System.out.println(Arrays.toString(arr));

}

//HQL group by and like example

query = session.createQuery("select e.name, sum(e.salary), count(e)"

+ " from Employee e where e.name like '%i%' group by e.name");

List<Object[]> groupList = query.list();

for(Object[] arr : groupList){

System.out.println(Arrays.toString(arr));

}

//HQL order by example

query = session.createQuery("from Employee e order by e.id desc");

empList = query.list();

for(Employee emp3 : empList){

System.out.println("ID Desc Order Employee::"+emp3.getId()+","+emp3.getAddress().getCity());

}

//rolling back to save the test data

tx.rollback();

//closing hibernate resources

sessionFactory.close();

}

}

Notice that I am using HQL for Select, Update and Delete operations. It also shows how to use HQL Join and HQL Aggregate functions. When I run above hql example program, we get following output.

May 22, 2014 1:55:37 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>

INFO: HCANN000001: Hibernate Commons Annotations {4.0.4.Final}

May 22, 2014 1:55:37 PM org.hibernate.Version logVersion

INFO: HHH000412: Hibernate Core {4.3.5.Final}

May 22, 2014 1:55:37 PM org.hibernate.cfg.Environment <clinit>

INFO: HHH000206: hibernate.properties not found

May 22, 2014 1:55:37 PM org.hibernate.cfg.Environment buildBytecodeProvider

INFO: HHH000021: Bytecode provider name : javassist

May 22, 2014 1:55:37 PM org.hibernate.cfg.Configuration configure

INFO: HHH000043: Configuring from resource: hibernate.cfg.xml

May 22, 2014 1:55:37 PM org.hibernate.cfg.Configuration getConfigurationInputStream

INFO: HHH000040: Configuration resource: hibernate.cfg.xml

May 22, 2014 1:55:37 PM org.hibernate.cfg.Configuration doConfigure

INFO: HHH000041: Configured SessionFactory: null

Hibernate Configuration loaded

Hibernate serviceRegistry created

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

WARN: HHH000402: Using Hibernate built-in connection pool (not for production use!)

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH000401: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost/TestDB]

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH000046: Connection properties: {user=pankaj, password=\*\*\*\*}

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH000006: Autocommit mode: false

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

INFO: HHH000115: Hibernate connection pool size: 20 (min=1)

May 22, 2014 1:55:37 PM org.hibernate.dialect.Dialect <init>

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

May 22, 2014 1:55:37 PM org.hibernate.engine.jdbc.internal.LobCreatorBuilder useContextualLobCreation

INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4

May 22, 2014 1:55:38 PM org.hibernate.engine.transaction.internal.TransactionFactoryInitiator initiateService

INFO: HHH000399: Using default transaction strategy (direct JDBC transactions)

May 22, 2014 1:55:38 PM org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory <init>

INFO: HHH000397: Using ASTQueryTranslatorFactory

Hibernate: select employee0\_.emp\_id as emp\_id1\_1\_, employee0\_.emp\_name as emp\_name2\_1\_, employee0\_.emp\_salary as emp\_sala3\_1\_ from EMPLOYEE employee0\_

Hibernate: select address0\_.emp\_id as emp\_id1\_0\_0\_, address0\_.address\_line1 as address\_2\_0\_0\_, address0\_.city as city3\_0\_0\_, address0\_.zipcode as zipcode4\_0\_0\_, employee1\_.emp\_id as emp\_id1\_1\_1\_, employee1\_.emp\_name as emp\_name2\_1\_1\_, employee1\_.emp\_salary as emp\_sala3\_1\_1\_ from ADDRESS address0\_ left outer join EMPLOYEE employee1\_ on address0\_.emp\_id=employee1\_.emp\_id where address0\_.emp\_id=?

Hibernate: select address0\_.emp\_id as emp\_id1\_0\_0\_, address0\_.address\_line1 as address\_2\_0\_0\_, address0\_.city as city3\_0\_0\_, address0\_.zipcode as zipcode4\_0\_0\_, employee1\_.emp\_id as emp\_id1\_1\_1\_, employee1\_.emp\_name as emp\_name2\_1\_1\_, employee1\_.emp\_salary as emp\_sala3\_1\_1\_ from ADDRESS address0\_ left outer join EMPLOYEE employee1\_ on address0\_.emp\_id=employee1\_.emp\_id where address0\_.emp\_id=?

Hibernate: select address0\_.emp\_id as emp\_id1\_0\_0\_, address0\_.address\_line1 as address\_2\_0\_0\_, address0\_.city as city3\_0\_0\_, address0\_.zipcode as zipcode4\_0\_0\_, employee1\_.emp\_id as emp\_id1\_1\_1\_, employee1\_.emp\_name as emp\_name2\_1\_1\_, employee1\_.emp\_salary as emp\_sala3\_1\_1\_ from ADDRESS address0\_ left outer join EMPLOYEE employee1\_ on address0\_.emp\_id=employee1\_.emp\_id where address0\_.emp\_id=?

Hibernate: select address0\_.emp\_id as emp\_id1\_0\_0\_, address0\_.address\_line1 as address\_2\_0\_0\_, address0\_.city as city3\_0\_0\_, address0\_.zipcode as zipcode4\_0\_0\_, employee1\_.emp\_id as emp\_id1\_1\_1\_, employee1\_.emp\_name as emp\_name2\_1\_1\_, employee1\_.emp\_salary as emp\_sala3\_1\_1\_ from ADDRESS address0\_ left outer join EMPLOYEE employee1\_ on address0\_.emp\_id=employee1\_.emp\_id where address0\_.emp\_id=?

List of Employees::1,San Jose

List of Employees::2,Santa Clara

List of Employees::3,Bangalore

List of Employees::4,New Delhi

Hibernate: select employee0\_.emp\_id as emp\_id1\_1\_, employee0\_.emp\_name as emp\_name2\_1\_, employee0\_.emp\_salary as emp\_sala3\_1\_ from EMPLOYEE employee0\_ where employee0\_.emp\_id=?

Employee Name=Lisa, City=Bangalore

Hibernate: select employee0\_.emp\_id as emp\_id1\_1\_, employee0\_.emp\_name as emp\_name2\_1\_, employee0\_.emp\_salary as emp\_sala3\_1\_ from EMPLOYEE employee0\_

Paginated Employees::1,San Jose

Paginated Employees::2,Santa Clara

Paginated Employees::3,Bangalore

Paginated Employees::4,New Delhi

Hibernate: update EMPLOYEE set emp\_name=? where emp\_id=?

Employee Update Status=1

Hibernate: delete from ADDRESS where emp\_id=?

Address Delete Status=1

Hibernate: delete from EMPLOYEE where emp\_id=?

Employee Delete Status=1

Hibernate: select sum(employee0\_.emp\_salary) as col\_0\_0\_ from EMPLOYEE employee0\_

Sum of all Salaries= 600.0

Hibernate: select employee0\_.emp\_name as col\_0\_0\_, address1\_.city as col\_1\_0\_ from EMPLOYEE employee0\_ inner join ADDRESS address1\_ on employee0\_.emp\_id=address1\_.emp\_id

[Pankaj Kumar, San Jose]

[David, Santa Clara]

[Lisa, Bangalore]

Hibernate: select employee0\_.emp\_name as col\_0\_0\_, sum(employee0\_.emp\_salary) as col\_1\_0\_, count(employee0\_.emp\_id) as col\_2\_0\_ from EMPLOYEE employee0\_ where employee0\_.emp\_name like '%i%' group by employee0\_.emp\_name

[David, 200.0, 1]

[Lisa, 300.0, 1]

Hibernate: select employee0\_.emp\_id as emp\_id1\_1\_, employee0\_.emp\_name as emp\_name2\_1\_, employee0\_.emp\_salary as emp\_sala3\_1\_ from EMPLOYEE employee0\_ order by employee0\_.emp\_id desc

ID Desc Order Employee::3,Bangalore

ID Desc Order Employee::2,Santa Clara

ID Desc Order Employee::1,San Jose

May 22, 2014 1:55:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop

INFO: HHH000030: Cleaning up connection pool [jdbc:mysql://localhost/TestDB]

Notice that once delete operation is performed, further operations are not showing that record data (sum of salary is 600). However I am rolling back the transaction, so the data in table will remain unchanged. Change the code to commit the transaction and it will be reflected in the database tables. I don’t like using HQL query a lot because as you can see that we need to take care of table mappings in our code. If we will use Session to delete the Employee object, it will delete the record from both the tables. You can download the sample hql example project from below link and try more examples.