HQL查询

HQL查询

- ✓ Hibernate提供了异常强大的查询体系，使用Hibernate有多种查询方式
- ✓ HQL(Hibernate Query Lanaguage)，语法很像SQL的语法，但是HQL是一种面向对象的语言，操作的对象是类，实例，属性。
- ✓ HQL查询依赖于Query类，每一个Query实例对应一个查询对象
- ✓ HQL语法不区分大小写，但是HQL语句中的包名，类名，实例名，属性名都区分大小写

HQL查询的步骤

1. 获取Session对象
2. 编写HQL语句
3. 以HQL语句作为参数，调用Session的createQuery方法创建查询对象
4. 如果HQL语句包含参数，调用Query的setXXX方法为参数赋值
5. 调用Query对象的list等方法遍历查询结果

查询单一属性

- ✓ 查询所有员工的姓名

```
//创建查询对象设定查询语句
Query query = session.createQuery("select ename from Emp");

//得到结果集封装到List集合，如果是查询一个列，元素类型就是列的类型
List<String> list = query.list();

//遍历集合中的元素
for (String ename : list) {
    System.out.println(ename);
}
```

查询单一属性

✓ 查询所有员工的编号

```
// 创建查询对象设定查询语句
Query query = session.createQuery("select id from Emp");

// 得到结果集封装到List集合，如果是查询一个列，元素类型就是列的类型
List<Integer> list = query.list();

// 遍历集合中的元素
for (Integer id : list) {
    System.out.println(id);
}
```

查询多个属性

- ✓ 查询员工的编号，姓名

```
Query query = session.createQuery("select id,ename from Emp");
List list = query.list();
for (Object obj : list) {
    Object[] emp = (Object[]) obj;
    System.out.println("编号：" + emp[0] + "，姓名：" + emp[1]);
}
```

查询多个属性

- ✓ 查询所有的员工编号，姓名，和部门名称

```
Query query = session.createQuery("select
    emp.id,emp.ename,emp.dept.dname from Emp emp");
//返回的集合中的元素都是Object[]数组类型
List list = query.list();
for (Object obj : list) {
    Object[] emp = (Object[]) obj;
    System.out.println("编号：" + emp[0] + "，姓名：" + emp[1] + "，部门：" +
    emp[2]);
}
```

查询多个属性

- ✓ 查询员工的编号，姓名，部门

```
Query query = session.createQuery("select
    emp.id,emp.ename,emp.dept.dname from Emp emp");
List list = query.list();
for (Object obj : list) {
    Object[] emp = (Object[]) obj;
    System.out.println("编号：" + emp[0] + "，姓名：" + emp[1] + "，部门：" +
    emp[2]);
}
```

利用构造方法把属性封装为实体对象

- ✓ 如果感觉Object[]数组不够对象化，可以利用构造方法实例化对象

```
String hql = "select new Emp(emp.id,emp.ename) from Emp emp";
Query query = session.createQuery(hql);
List<Emp> list = query.list();
for (Emp emp : list){
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

实体对象查询

- ✓ 查询所有的员工的编号和姓名

```
String hql = "from Emp emp";
Query query = session.createQuery(hql);
List<Emp> list = query.list();
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

实体对象查询

- ✓ 如果配置了多对一的关系，还可以查询所有的员工的编号，姓名，部门名称

```
String hql = "from Emp emp";
Query query = session.createQuery(hql);
List<Emp> list = query.list();
for (Emp emp : list) {
    System.out.println("编号：" + emp.getEmpno());
    System.out.println("姓名：" + emp.getEname());
    //注意这个判断，因为员工的外键可能为null
    if (emp.getDept() != null) {
        Dept dept = emp.getDept();
        System.out.println("部门：" + dept.getDname());
    }
    System.out.println("----------------------------");
}
```

实体对象查询

- ✓ 如果配置一对多的关系，还可以查询所有的部门编号，部门名称，以及部门所有的员工

```
String hql = "from Dept dept";
List<Dept> list = session.createQuery(hql).list();
for (Dept dept : list) {
    System.out.println(dept.getDeptno() + "," + dept.getDname());
    //如果员工集合存在数据，则遍历员工数据
    if (dept.getEmps() != null && dept.getEmps().size() != 0) {
        List<Emp> emps = dept.getEmps();
        for (Emp emp : emps) {
            System.out.println("\t\t" + emp.getEmpno() + "," + emp.getEname());
        }
    }
}
```

条件查询

- ✓ 查询员工编号小于5的员工信息，返回实体对象

```
//"select e"可以不写
String hql = "select e from Emp e where e.empno<5";
List<Emp> list = session.createQuery(hql).list();
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

带占位符?的条件查询

- ✓ 查询员工编号小于5并且部门编号等于30的员工信息

```
String hql = "select e from Emp e where e.empno<? and e.dept.deptno=?";
Query query = session.createQuery(hql);
//给占位符赋值，从0开始
query.setParameter(0, 5);
query.setParameter(1, 30);
List<Emp> list = query.list();
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

带命名参数的条件查询

- ✓ 查询员工编号小于5并且部门编号等于30的员工信息，注意命名参数用:开头

```
String hql = "select e from Emp e where e.empno<:empno and
    e.dept.deptno=:deptno";
Query query = session.createQuery(hql);
// 给命名参数赋值
query.setParameter("empno", 5);
query.setParameter("deptno", 30);
List<Emp> list = query.list();
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

给命名参数赋集合或者数组类型的条件查询

✓ 查询员工编号是1，3，5的员工信息

```
String hql = "select e from Emp e where e.empno in (:myids)";
Query query = session.createQuery(hql);
// 给命名参数赋一个数组
query.setParameterList("myids", new Object[]{1,3,5});
List<Emp> list = query.list();
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

分页查询

- ✓ 分页查询主要是靠Query对象的两个方法实现的
  - ➢ setFirstResult(N)：从第N条开始，注意N从0开始
  - ➢ setMaxResults(N)：每页N条

```
Query query = session.createQuery("from Emp e order by e.empno");
query.setFirstResult(2);//从第3条开始
query.setMaxResults(4);//返回4条数据
List<Emp> list = query.list();//
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

内连接查询

✓ 查询员工编号，姓名，部门编号，部门名称

```
String hql = "select e.empno,e.ename,d.deptno,d.dname from Emp e join
    e.dept d";
Query query = session.createQuery(hql);
List list = query.list();
for (Object obj : list) {
    Object[] data = (Object[]) obj;
    System.out.println("员工编号：" + data[0]);
    System.out.println("员工姓名：" + data[1]);
    System.out.println("部门编号：" + data[2]);
    System.out.println("部门名称：" + data[3]);
    System.out.println("----------------------");
}
```

左外连接查询

- ✓ 查询员工编号，姓名，部门编号，部门名称，没有部门的员工部门编号和部门名称会为空

```
String hql = "select e.empno,e.ename,d.deptno,d.dname from Emp e left join
    e.dept d";
Query query = session.createQuery(hql);
List list = query.list();
for (Object obj : list) {
    Object[] data = (Object[]) obj;
    System.out.println("员工编号：" + data[0]);
    System.out.println("员工姓名：" + data[1]);
    System.out.println("部门编号：" + data[2]);
    System.out.println("部门名称：" + data[3]);
    System.out.println("----------------------");
}
```

右外连接查询

- ✓ 查询员工编号，姓名，部门编号，部门名称，没有员工的部门的员工编号和员工姓名会为空

```
String hql = "select e.empno,e.ename,d.deptno,d.dname from Emp e right join
    e.dept d";
Query query = session.createQuery(hql);
List list = query.list();
for (Object obj : list) {
    Object[] data = (Object[]) obj;
    System.out.println("员工编号：" + data[0]);
    System.out.println("员工姓名：" + data[1]);
    System.out.println("部门编号：" + data[2]);
    System.out.println("部门名称：" + data[3]);
    System.out.println("----------------------");
}
```

统计查询

- ✓ 查询员工的人数

```
String hql = "select count(*) from Emp";
//查询返回的集合中只有一个元素，是Long型
List list = session.createQuery(hql).list();
Long empcount = (Long) list.get(0);
System.out.println("员工人数：" + empcount);
```

- ✓ 也可以写成单一值写法

```
String hql = "select count(*) from Emp";
//返回一个单一值
Long empcount = (Long) session.createQuery(hql).uniqueResult();
System.out.println("员工人数：" + empcount);
```

统计查询2

✓ 查询多个统计结果：员工人数和编号总和

```
String hql = "select count(*),sum(empno) from Emp";
//查询返回的集合中只有一个元素，是Object数组
List list = session.createQuery(hql).list();
Object[] datas = (Object[]) list.get(0);
Long empcount = (Long) datas[0];
Long empnosum = (Long) datas[1];
System.out.println("员工人数：" + empcount + "，编号总和：" + empnosum);
```

统计查询3

✓ 分组查询：查询每个部门的部门名称和部门人数

```
String hql = "select d.dname,count(e.empno) from Emp e right join e.dept d
    group by d.dname";
List list = session.createQuery(hql).list();
for (Object obj : list) {
    Object[] datas = (Object[]) obj;
    System.out.println("部门：" + datas[0] + "，人数：" + datas[1]);
}
```

DML风格的查询

- ✓ 删除员工编号小于等于3的员工信息

```
String hql = "delete from Emp e where e.empno<=?";
Query query = session.createQuery(hql);
query.setParameter(0, 3);
int result = query.executeUpdate();
System.out.println("删除了" + result + "条员工信息");
```

QBC查询

- ✓ QBC查询就是通过使用Hibernate提供的Query By Criteria API来查询对象，这种API封装了SQL语句的动态拼装，对查询提供了更加面向对象的功能接口

QBC查询示例

- ✓ 查询姓名叫猪八戒的员工

```
//创建Criteria对象
Criteria criteria = session.createCriteria(Emp.class);
//增加查询条件ename='猪八戒'
criteria.add( Expression.eq("ename", "猪八戒") );
//返回集合
List<Emp> list = criteria.list();
//遍历集合
for (Emp emp : list) {
    System.out.println(emp.getEmpno() + "," + emp.getEname());
}
```

抓取策略

- ✓ 抓取策略可以用在以下两种情况
  - ➢ 单端关联 例如<many-to-one> <one-to-one>标签中
  - ➢ 集合 例如<set> <bag>标签中
- ✓ 抓取策略用fetch属性设置，有两个取值
  - ➢ fetch="select" 表示单独发出一条查询语句抓取当前对象关联的实体或者是集合，这是默认设置
  - ➢ fetch="join" 表示发出一条外连接语句来抓取当前对象关联的实体或者是集合，此时lazy会失效

抓取策略配置示例

- ✓ <many-to-one name="dept" column="deptno" fetch="join"/>
- ✓ <set name="emps" fetch="join">
- ✓ <bag name="emps" fetch="join">