Hibernate

Desenvolvimento Web

HQL
- Hibernate Query Language -

Introdução

- Hibernate Query Language (HQL) é uma linguagem de consulta orientada a objetos
- Ao invés de operar sobre tabelas e colunas, HQL trabalha com objetos persistentes e suas propriedades
- Consultas HQL são traduzidas pelo Hibernate em consultas SQL convencionais, as quais são executadas no banco de de dados relacional.

Cláusula FROM

Define/seleciona a classe sobre a qual a consulta será executada

```
String hql = "FROM Employee";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula AS

Usada para definir apelidos (aliases) para nomes de classe usados em consultas HQL

```
String hql = "FROM Employee AS E";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula SELECT

Define o conjunto de propriedades dos objetos selecionados em consultas HQL

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

Cláusula WHERE

Define opções/condições que serão aplicadas sobre os objetos selecionados a fim de filtrá-los

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

Cláusula ORDER BY

Ordena os resultados de uma consulta HQL levando-se em consideração:

A propriedade a ser ordenada e o critério de ordenação (ASC or DESC)

```
String hql = "FROM Employee E WHERE E.id > 10 ORDER BY
    E.salary DESC";
Query query = session.createQuery(hql);
List results = query.list();
```

Cláusula GROUP BY

Agrupa informações com base no valor de um atributo especificado como parâmetro.

```
String hql = "SELECT SUM(E.salary), E.firtName FROM
    Employee E " + "GROUP BY E.firstName";
Query query = session.createQuery(hql);
List results = query.list();
```

Usando Parâmetros Nomeados

Permite que sejam definidas consultas HQL parametrizadas.

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

Métodos de Agregação

Função	Descrição
Avg (nome da propriedade)	A média dos valores de uma propriedade
Count (nome da propriedade)	O número de vezes que uma propriedade ocorre nos resultados
Max (nome da propriedade)	O valor máximo dentre os valores de uma propriedade
Min (nome da propriedade)	O valor mínimo dentre os valores de uma propriedade
Sum (nome da propriedade)	A soma dos valores de uma propriedade

```
String hql = "SELECT count(distinct E.firstName) FROM Employee E";
Query query = session.createQuery(hql);
List results = query.list();
```

Usando a API Criteria

Introdução

- A API Criteria permite a construção de consultas programaticamente
- Exemplo:

```
Criteria cr = session.createCriteria(Employee.class);
List results = cr.list();
```

Restrições com Criteria

```
...
// To get records having salary equals to 2000
cr.add(Restrictions.eq("salary", 2000));
// To get records having salary more than 2000
cr.add(Restrictions.gt("salary", 2000));
// To get records having salary less than 2000
cr.add(Restrictions.lt("salary", 2000));
// To get records having fistName starting with zara
cr.add(Restrictions.like("firstName", "zara%"));
// Case sensitive form of the above restriction.
cr.add(Restrictions.ilike("firstName", "zara%"));
// To get records having salary in between 1000 and 2000
cr.add(Restrictions.between("salary", 1000, 2000));
// To check if the given property is null
cr.add(Restrictions.isNull("salary"));
// To check if the given property is not null
cr.add(Restrictions.isNotNull("salary"));
// To check if the given property is empty
cr.add(Restrictions.isEmpty("salary"));
// To check if the given property is not empty
cr.add(Restrictions.isNotEmpty("salary"));
...
```

Operadores AND e OR

```
Criteria cr = session.createCriteria(Employee.class);
Criterion salary = Restrictions.gt("salary", 2000);
Criterion name = Restrictions.ilike("firstNname","zara%");

// To get records matching with OR condistions
LogicalExpression orExp = Restrictions.or(salary, name);
cr.add( orExp );

// To get records matching with AND condistions
LogicalExpression andExp = Restrictions.and(salary, name);
cr.add( andExp );

List results = cr.list();
```

Ordenando Resultados

```
Criteria cr = session.createCriteria(Employee.class);
// To get records having salary more than 2000
cr.add(Restrictions.gt("salary", 2000));

// To sort records in descening order
cr.addOrder(Order.desc("salary"));

// To sort records in ascending order
cr.addOrder(Order.asc("salary"));

List results = cr.list();
```

Agregações/Projeções

```
Criteria cr = session.createCriteria(Employee.class);


// To get total row count.
cr.setProjection(Projections.rowCount());
// To get average of a property.
cr.setProjection(Projections.avg("salary"));
// To get distinct count of a property.
cr.setProjection(Projections.countDistinct("firstName"));
// To get maximum of a property.
cr.setProjection(Projections.max("salary"));
// To get minimum of a property.
cr.setProjection(Projections.min("salary"));
// To get sum of a property.
cr.setProjection(Projections.sum("salary"));
```

Bibliografia

- http://www.tutorialspoint.com/hibernate/hibernate_query_language.htm
- http://www.tutorialspoint.com/hibernate/hibernate_criteria_queries.htm