Diseño de Sistemas Software

HQL
Hibernate Query Language

Índice

Introducción

- *Hibernate Query Language* (HQL) es un lenguaje similar a SQL.
- No opera sobre las tablas de la base de datos y sus columnas sino sobre los objetos de Hibernate y sus propiedades.
- Por lo tanto es independiente de la base de datos.
- Lenguaje fácil de aprender ya que es una mezcla de SQL y Java.
- Los operadores de SQL como SELECT, FROM, WHERE etc. no son sensibles a mayúsculas, pero las propiedades y el nombre de los objetos sí lo son.
- Utiliza el mecanismo de búsqueda createQuery de Hibernate.

Tipos de asociación

- HQL soporta 2 tipos de asociación (join):
  - **Implícita**: En la sentencia no se usa la palabra JOIN si la relación está presente en el objeto. Solo funciona cuando se navega por relaciones con cardinalidad máxima 1.

    Clases EN:

```
Class LineaPedido {
    private int idLineaPedido;
    private int cantidad;
    private Articulo articulo;
}
```

```
Class Articulo {
   private int idArticulo;
   private String nombre;
}
```

- **HQL**: from LineaPedido as lp where **lp.Articulo.Nombre** like ‘s%’
- **Resultado**: Se obtienen todas las líneas que contengan un articulo que empiece por “s”.

Tipos de asociación

- **Explicita**: Hay 3 tipos de join's. P.e. dadas 2 entidades E1 y E2 donde ambas están relacionadas, tendríamos los siguientes casos:
  - **inner join**: la unión de las instancias de E1 presentes en E2 y viceversa (igual al join implícito).
  - **left outer join**: las instancias de E1 que estén presentes o NO (a null) en E2 y las de E2 que estén presentes en E1.
  - **right outer join**: las instancias de E1 que estén presentes en E2 y las de E2 que estén presentes o NO (a null) en E1.
- El uso explícito es imprescindible cuando recorremos relaciones con cardinalidad > 1 (1:M o M:M)
  - **Ejemplo**: Un artículo se relaciona con muchas líneas (0..*):
    - ***HQL**: "select art from Articulo as art **inner join** art.Lineas as linea where linea.Id = 12"*
    - ***Resultado**: Obtengo todos los artículos que apunten a una linea con id igual a 12*

Clausula Where

- La clausula **where** permite crear condiciones que filtran la colección de objetos de una clase.
  - **HQL**: “select articulo from Articulo as articulo inner join articulo.Lineas as linea **where linea.Cantidad>5**”
  - **Resultado**: Obtengo los artículos cuyas líneas contengan una cantidad superior a 5
- **Cuidado!!**: HQL es *case-sensitive* para las relaciones y propiedades y estas comienzan siempre con la primera letra en mayúsculas en todos los ENs.

Clausula Select

- La cláusula **select** escoge qué objetos y propiedades devolver del conjunto de resultados de la consulta

- Si realizamos un operador explicito (join) se obtienen 2 colecciones de objetos, se devuelve entonces un array de Objects, con select podemos indicar que tipo de objetos queremos devolver

- En la siguiente consulta solo devolvemos clientes:

  Select cli from Cliente as cli inner join cliente.Pedido as pedido where pedido.Id = 1

Clausula Select

- Permite utilizar los siguientes funciones de agregación sobre propiedades (de tipo numérico):

  avg(...), sum(...), min(...), max(...)

- El operador **Count**: Devuelve el número de objetos de una colección, por ejemplo:

  Select **Count (cli)** from Cliente as cli where cli.Nombre like ‘s%’

Consultas polimórficas con HQL

- HQL soporta la herencia definida sobre los objetos EN
- Si tenemos una jerarquía de Usuario del que heredan Administrador y UsuarioWeb. Una consulta como esta:

```
from Usuario as usu
```

  - Devuelve **todas** las instancias de Usuario, **incluidas** las instancias de las clases que heredan de usuario como Administrador y UsuarioWeb
- Las consultas devolverán instancias de las clases persistentes que extiendan de dicha clase o implementen su interfaz.

Herencia con HQL

- La propiedad especial **class** accede al valor discriminador de una instancia en el caso de la persistencia polimórfica.

- Siguiendo el ejemplo del caso anterior, podemos indicar en la clausula **where** que queremos solo las instancias de un determinado subtipo:

  From Usuario usu where **usu.class =** UsuarioWeb

Tratamiento de Colecciones con HQL

- **Size**: Obtenemos el tamaño de una propiedad de tipo colección.

  From Cliente cli where **cli.Pedidos.size** > 0

- O Alternativamente

  From Cliente cli where **size(cli.Pedidos)** > 0

Tratamiento de Colecciones con HQL

- Podemos comprobar que un elemento se encuentra o no en una determinada colección con los operadores:

  member of and **not member of**

- Ejemplo **member of**:

  select cli from Cliente cli, Registro reg where cli **member of** reg.ClientesMorosos

- Ejemplo de **not member of**:

  select cli from Cliente cli, Registro reg where cli **not member of** reg.ClientesMorosos

Tratamiento de Colecciones con HQL

- Los operadores **exists, all, in** tienen que combinarse con **elements** para poder aplicarse sobre los objetos de una colección:
  - **Exists**: Devuelve verdadero si existen elementos en dicha colección

    From Cliente c where **exists elements**(c.Order)

  - **All**: Devuelve verdadero cuando todos los elementos cumplen la condición comparada

    From Player p where 3 > **all elements**(p.Scores)

  - **In**: Devuelve verdadero si el objeto comparado está contenido por dicha colección

    From Cliente c where '34349494' **in elements**(c.Phones)

    P.e. relación M:M: From Recetas r where :usuario in (from r.EsRecetaFav)

Tratamiento de Fechas

- Operadores para convertir un objeto tipo Date, Time o TimeStamp en los siguientes:
  - second(...), minute(...), hour(...),
  - day(...), month(...), year(...)
- Por ejemplo:

  from Pedido p where **year**(p.FechaEnvio) > 2007

Resto de Operadores

- Operadores Matemáticos: **+, -, *, /**
- Operadores Binarios de comparación:

  =, >=, <=, <>, !=, like

- Operadores Lógicos: **and, or, not**
- Operadores de Cadenas:
  - Contatenación: **|| or concat()**
  - Subcadenas: **substring(), trim()**,
  - Mayusculas y minúsculas: **lower(), upper()**
  - Longitud: **length()**

Comprobación de Existencia

- En el caso de las propiedades **univaluadas** contamos con 2 operadores:
  - **Is null** (Es nulo) e **is not null** (no es nulo)

    From Pedido p where p.Descripcion **is not null**

- En el caso de las propiedades **multivaluadas** tenemos los operadores:
  - **is empty**: comprueba que esté vacía la colección
  - **is not empty**: comprueba que no esté vacía la colección

    From Pedido p where p.LineasPedido **is empty**

Paso de parámetros

- Queremos filtrar por un valor que recibimos como **parámetro**.

- Para ello, los parámetros se indican en HQL poniendo **“:”** delante.

From Cliente cli where cli.Ciudad = **:p_ciudad**

- En OOH4RIA podemos usar HQL en las operaciones tipo **ReadFilter**:
  - Desde el modelado gráfico podemos introducir la query HQL directamente en el campo **Filter** de la sección Properties.
  - Desde el modelado textual se realiza usando la siguiente notación:

```
public readFilter::ReadFilter(p_nombre : String):List<Object->Cliente>
{filter="From ClienteEN c where c.Nombre = :p_nombre"};
```

Referencia HQL

- Más Información sobre HQL:

  http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html

Ejercicio

- Añade las siguientes operaciones al modelo creado en las secciones anteriores (gráfico o textual):
  - buscarClientePorNombre(...)
  - dameClientesConPedidos()
  - dameClientesConMasDeNPedidos(...)
  - damePedidosDelAnyo(...)
  - damePedidosEnEstado(...)
  - buscarProductosConNombreQueEmpiecePor(...)
  - dameProductosSinStock()