HIBERNATE

Sesión 4: Consultas. HQL

Puntos a tratar

- Consultas y procesamiento de resultados
- HQL: Hibernate Query Language
- SQL nativo

Consultas con Hibernate

- Si no conocemos los identificadores de los objetos que estamos buscando, necesitamos realizar una *query* (operación de consulta a la base de datos)
- Las consultas se representan mediante `org.hibernate.Query` a través de la sesión actual
  - Ej. `List query = session.createQuery("...").list();`
  - El resultado de la consulta son instancias persistentes que se "cargan" en memoria

Procesamiento de resultados

- Iteración de resultados: Métodos *list()* o *iterate()*

```
Iterator pujas = session.createQuery(
                 "from Puja as puja where puja.fecha < ?")
                .setDate(0, date) .
                .iterate(); //o list()
while (pujas.hasNext() ) {
     Puja puja = (Puja) pujas.next();
     //hacer algo que no podemos expresar con la consulta
     ...
}
```

- El método *iterate()* se utiliza en lugar de list() para mejorar el rendimiento, en el caso de que las instancias devueltas por la consulta ya estén en la sesión.

Consultas que devuelven tuplas

- En una consulta "join" se devuelven tuplas (*arrays*) de objetos.

```
Query q = session.createQuery(
        "select a from Articulo a join a.pujas p");
Iterator pares = q.list().iterator();
while (pares.hasNext() ) {
    Object[] pares = (Object[]) pares.next();
    Articulo art = (Articulo) pares[0];
    Puja puja = (Puja) pares[1];
}
```

Resultados escalares

- Las propiedades o los resultados de funciones "agregadas" se consideran resultados escalares.

```
Iterator results = sess.createQuery(
                "select articulo.color, min(articulo.fechaAlta),"
                + "count(articulo) from Articulo articulo "
                + "group by articulo.color")
               .list()
               .iterator();
while ( results.hasNext() ) {
      Object[] row = (Object[]) results.next();
      Color type = (Color) row[0];
      Date oldest = (Date) row[1];
      Integer count = (Integer) row[2];
      ...
}
```

Enlazado (*binding*) de parámetros

- Se pueden enlazar valores a parámetros con nombre. Los parámetros se numeran comenzando por cero.

```
Query q = sess.createQuery(
            "from Articulo art where art.name = :name");
q.setString("name", "Cuadro");
Iterator arts = q.iterate();

//lista de parámetros con nombre
List nombres = new ArrayList();
nombres.add("Collar");
nombres.add("Copa");
Query q = sess.createQuery(
      "from Articulo art where art.name in (:listaNombres)");
q.setParameterList("listaNombres", nombres);
List articulos = q.list();
```

Paginación

- Podemos especificar límites sobre el conjunto de resultados

```
Query q = sess.createQuery("from Articulo art");
q.setFirstResult(20);
q.setMaxResults(10);
List articulos = q.list();
```

Recuperación de objetos persistentes

- Hibernate proporciona varias formas de recuperar objetos de la BD:
  - Navegando por el grafo de objetos

```
user.getDirection.getCiudad();
```

  - A través de su identificador

```
Usuario user = (Usuario) session.get(Usuario.class, new
  Long(userID);
```

  - Mediante HQL
  - Utilizando el API `Criteria` (no necesita manipular *Strings*)
  - Haciendo uso de consultas SQL nativas

HQL: Hibernate Query Language

- HQL es un lenguaje de consultas orientado a objetos
  - HQL sólo se utiliza para recuperar datos.
- Estructura de una consulta HQL

  [select ...] from ... [where ...] [group by ... [having ...]] [order by ...]
- Las consultas se representan con una instancia de la interfaz `Query`
  - Preparamos la consulta (posiblemente con parámetros)
  - Asignamos los valores de los parámetros:

    setString(), setDate(), setInteger,...
  - Realizamos la consulta:

    list(), iterate()

Uso de HQL

- Utilizando literales de cadenas de caracteres:

```
Query hqlQuery =
session.createQuery("from Usuario u where u.nombre= :fname");
hqlQuery.setString("fname","Pepe");
hqlQuery.setFirstResult(0);
hqlQuery.setMaxResults(10); List result = hqlQuery.list();
```

- Utilizando *named queries:*

```
Query hqlQuery = session.getNamedQuery("findArticulosByDescripcion")
 .setString( "desc", descripcion).list() ;
```

  - En *Articulo.hbm.xml* tendríamos:

```
<query name="findArticulosByDescripcion">
    <from Articulo where articulo.descripcion like :desc>
</query>
```

Consultas básicas y restricciones

- *from*: recupera objetos

```
from Puja
from Puja as puja
```

- restricciones (*where*)

```
from Usuario u where u.email = 'usuario@hibernate.org'
from Puja puja where puja.cantidad between 1 and 10
from Puja puja where puja.cantidad > 100
from Usuario u where u.email is not null
from Usuario u where u.nombre like "G%"
from Usuario user
      where (user.nombre like "G%" and user.apellidos like "K%")
      or user.email in ("usu1@hibernate.org", "usu2@hibernate.org")
```

Más consultas básicas

- Ordenación de resultados (*order by*)

```
from Usuario u order by u.apellidos asc, u.nombre asc
```

- Uso de agregación (*count()*, *min()*, *max()*, *sum()*, *avg()*)

```
Integer count = (Integer) session
                .createQuery("select count(*) from Articulo")
                .uniqueResult();
```

- Agrupación de resultados (*group by*)

```
select puja.articulo.id, avg(puja.cantidad)
        from Puja puja group by puja.articulo.id
```

Estrategias de *fetching*

- Una *fetching strategy* es la estrategia que Hibernate utiliza para recuperar objetos asociados si la aplicación necesita navegar por la asociación. Las estrategias *fetch* pueden ser declaradas en el fichero de mapeado O/R.
  - *Join fetching (eager fetching), utiliza outer join* para recuperar la entidad asociada o colección
  - *Select fetching*, se utiliza un segundo SELECT para recuperar la entidad asociada o colección
  - *Lazy collection fetching*: opción por defecto para colecciones

Uso de *join*

- Usaremos la palabra reservada *join* para combinar datos de dos (o más) tablas:
  - Primero se realiza el producto cartesiano
  - Luego se filtran los datos mediante alguna condición
- Ejemplo:

ARTICULO

ARTIC_ID	NOMBRE	PRECIO_INICIAL
1	Cuadro	2.00
2	Mesa	50.00
3	Silla	1.00

PUJA

PUJA_ID	ARTIC_ID	CANTIDAD
1	1	10.00
2	1	20.00
3	2	55.00

from ARTICULO A left outer join PUJA P on A.ARTIC_ID = P. ARTIC_ID

ARTIC_ID	NOMBRE	PRECIO_INICIAL	PUJA_ID	ARTIC_ID	CANTIDAD
1	Cuadro	2.00	1	1	10.00
1	Cuadro	2.00	2	1	20.00
2	Mesa	50.00	3	2	55.00
3	Silla	1.00	null	null	null

Formas de expresar *join* (I)

- Un *join ordinario* en la cláusula from

```
Query q = session.createQuery("from Articulo ar join ar.pujas puja");
Iterator pairs = q.list().iterator();
while (pairs.hasNext() ) {
    Object[] pair = (Object[]) pairs.next();
    Articulo ar = (Item) pair[0];
    Puja puja = (puja) pair[1]; }
```

- Un *fetch join* en la cláusula from

```
from Articulo ar left join fetch ar.pujas where ar.descripcion like '%es%'
```

Formas de expresar *join* (II)

- Un *theta-style join* en la cláusula where (cuando entre dos clases no hay definida una asociación)

```
Iterator i = session.createQuery("from User user, LogRecord log "
                + "where user.username = log.username").list.iterator();
while (i.hasNext() ) {
    Object[] pair = (Object[]) i.next();
   User user = (User) pair[0];
   LogRecord log = (LogRecord) pair[1]; }
```

- Un *join* de asociación *implícito en las cláusulas where o select*

```
from Puja puja where puja.articulo.descripcion like '%es%'
```

SQL nativo

- Mediante la interfaz *SQLQuery* :
  - Session.createSQLQuery()

```
List articulos = sess.createSQLQuery(
        "select {art.*} from ARTICULOS art"
      + "were NOMBRE like 'Portatil%'")
     .addEntity("art",Articulo.class).list();
```

  - El método *addEntity()* asocia el alias SQL de la tabla con la clase entidad devuelta, y determina el contenido del *result set* de la consulta

sentencias SQL con nombre

- En fichero de mapeado (*.hbm.xml):

```
<sql-query name="personas">
   SELECT person.NAME, person.AGE, person.SEX
      FROM PERSON person
    WHERE person.NAME LIKE :namePattern </sql-query>
<sql-query name="findArticulosByDescripcion">
   select {a.*} from ARTICULO where DESCRIPCION like :desc
   <return alias="a" class="Articulo"/> </sql-query>
```

- Código Java asociado:

```
List articulos = session.getNamedQuery("findArticulosByDescripcion")
                .setString( "desc", descripcion)
                .list();
List compradores = sess.getNamedQuery("personas")
                .setString("namePattern", "David")
                .setMaxResults(50) .list();
```

¿Preguntas...?