Metodologías de Diseño y Programación
CC3002 @ 2009

Instructivo de Persistencia Utilizando Hibernate

Daniel Perovich – Andrés Vignaga
{dperovic, avignaga}@dcc.uchile.cl

Diploma en Ingeniería de Software
Departamento de Ciencias de la Computación
Universidad de Chile

Introducción

El presente instructivo describe los elementos mínimos necesarios para obtener soporte de persistencia utilizando Hibernate. Los ejemplos están basados en el sistema Point-of-Sale (POS). Por mayor detalle acerca del uso de Hibernate referirse a http://www.hibernate.org

Anotaciones

Se describe a continuación las anotaciones necesarias en el código fuente.

Clases

Las clases cuyas instancias **no** se desea persistir, **no** llevan anotaciones.

Las clases cuyas instancias **sí** se desea persistir, deben:

a) Importar el package *javax.persistence*

b) Utilizar la anotación *@Entity*

c) Declarar un atributo que será utilizado por Hibernate como identificador de instancias. Es recomendable que este atributo se de tipo primitivo, preferentemente *long*, y que tenga el mismo nombre en todas las clases a persistir, e.g. *id*. Este atributo **no** debe ser inicializado en el constructor, y **no** debe ser consultado o actualizado en ninguna parte del código.

d) Declarar el getter y setter para el atributo identificador, utilizando la anotación *@Id* (la anotación es *@Id* independientemente del nombre del atributo). Indicar además que la generación del valor para el atributo debe ser realizada en forma automática usando *@GeneratedValue(strategy=GenerationType.AUTO)*.

A continuación se presenta como ejemplo la adecuación de la clase *Product* para proveer soporte de persistencia. No olvidar registrar el nombre de la clase en el archivo de configuración de Hibernate (ver sección Configuración de Hibernate más adelante).

```
package pos.logic;

import javax.persistence.*;

@Entity
public class Product
{
        private long id;

        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        public long getId()
        {
                return this.id;
        }

        protected void setId(long id)
        {
                this.id = id;
        }

        ...
}
```

(a) (b) (c) (d)

Atributos

En aquellas clases cuyas instancias se desea persistir, todos los atributos deben tener getter y setter asociados. Los getters y setters deben tener, por lo menos, visibilidad *protected* (por lo tanto *public* también es correcto). Los atributos, getters y setters **no** necesitan llevar anotaciones.

Pseudo-atributos

En aquellas clases cuyas instancias se desea persistir, la definición de los pseudo-atributos debe cumplir las siguientes consideraciones.

a) El tipo del pseudo-atributo debe ser una clase persistente (marcada con *@Entity*) o una colección de una clase persistente.

b) Los pseudo-atributos correspondientes a colecciones, deben ser definidos utilizando interfaces genéricas; e.g. *Set<T>*, *List<T>* o *Map<K,T>*, siendo *T* el tipo de elementos a contener en la colección, y *K* el tipo de la propiedad que sea llave de *T*.

c) Los pseudo-atributos correspondientes a colecciones pueden ser inicializados con cualquier colección que realice la interfaz, e.g. *HashSet<T>*, *ArrayList<T>* o *HashMap<K,T>*.

d) Debe proveerse un getter y un setter para cada pseudo-atributo.

e) Los getters deben anotarse usando una de las cuatro combinaciones de relación entre objetos, las cuales dependen de las multiplicidades de la asociación, a saber *@OneToOne*, *@OneToMany*, *@ManyToOne* y *@ManyToMany*.

f) Todas las anotaciones en los getters deben incluir *cascade=CascadeType.ALL* como argumento.

g) Para el caso especial de pseudo-atributos de tipo *Map<K,T>*, debe indicarse qué propiedad de *T* es llave, utilizando la anotación *@MapKey(name="propname")*.

A continuación se presenta como ejemplo la adecuación de la clase *Sale*.

```
package pos.logic;

import javax.persistence.*;

@Entity
public class Sale
{
        // Attributes
        private long id;

        private int number;
        private Date date;

        private List<Item> items;

        private Payment payment;


        // Constructors
        public Sale()
        {
                this.items = new ArrayList<Item>();
                this.payment = null;
        }
```

(b) *items* declarado usando la interfaz *List<Item>*

(a) Ambas clases, Item y Payment, deben ser declaradas usando *@Entity* (esto se encuentra en Item.java y Payment.java)

(c) Colección *item* inicializada utilizando la clase *ArrayList<Item>* que realiza la interfaz *List<Item>*

```
        // Properties
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        public long getId()
        {
                return this.id;
        }

        protected void setId(long id)
        {
                this.id = id;
        }

        public int getNumber()
        {
                return this.number;
        }

        protected void setNumber(int number)
        {
                this.number = number;
        }

        public Date getDate()
        {
                return this.date;
        }

        protected void setDate(Date date)
        {
                this.date = date;
        }

        @OneToMany(cascade=CascadeType.ALL)
        public List<Item> getItems()
        {
                return this.items;
        }

        protected void setItems(List<Item> items)
        {
                this.items = items;
        }

        @OneToOne(cascade=CascadeType.ALL)
        public Payment getPayment()
        {
                return this.payment;
        }

        protected void setPayment(Payment payment)
        {
                this.payment = payment;
        }

        ...
}
```

(d) Getter y setter para cada pseudo-atributo, *items* y *payment*

(e) El getter anotado según la multiplicidad de cada pseudo-atributo: *@OneToMany* para *items* y *@OneToOne* para *payment*

(f) Ambas anotaciones llevan *cascade=CascadeType.ALL* como argumento

A continuación se presenta como ejemplo la clase Catalog que incluye un pseudo-atributo que utiliza un map.

```
package pos.logic;

import java.util.*;

import javax.persistence.*;

@Entity
public class Catalog
{
        // Attributes
        private long id;

        private Map<Long,Product> products;

        // Constructors
        public Catalog()
        {
                this.products = new HashMap<Long,Product>();
        }

        // Properties
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        public long getId()
        {
                return this.id;
        }

        protected void setId(long id)
        {
                this.id = id;
        }

        @OneToMany(cascade=CascadeType.ALL)
        @MapKey(name="code")
        public Map<Long,Product> getProducts()
        {
                return this.products;
        }

        protected void setProducts(Map<Long,Product> pro
        {
                this.products = products;
        }

        ...
}
```

(b) *products* declarado usando la interfaz *Map<Long,Product>*

(c) *products* inicializado utilizando la clase *HashMap<Long,Product>* que realiza la interfaz *Map<Long,Product>*

(e) El getter anotado según la multiplicidad del pseudo-atributo: *@OneToMany* para *products*

(f) La anotación lleva *cascade=CascadeType.ALL* como argumento

(g) Se indica que el atributo *code* de la clase *Product* es la llave para el map, usando para ello *@MapKey(name="code")*

Configuración de Hibernate

El archivo de configuración de Hibernate (*hibernate.cfg.xml*) debe estar localizado en la raíz del classpath de Java. Si se está utilizando Eclipse, es suficiente con ubicar el archivo en el directorio con los fuentes (*/src*). Si tal archivo no existe, crearlo copiando el del ejemplo del POS.

El archivo de configuración (*hibernate.cfg.xml*) debe listar todas las clases que contienen anotaciones. Por ejemplo, dada la clase *Product* en el package *pos.logic*, debe agregarse la línea <*mapping class="pos.logic.Product"*/> al archivo de configuración. Esto debe hacerse por cada clase cuyas instancias se desea persistir. A continuación se presenta el ejemplo completo para el caso del POS. Tener presente que la parte a editar por parte de un desarrollador es la final, donde se detallan los mapeos. El resto del código, es decir, la parte inicial, es autogenerado.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
       "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">org.hsqldb.jdbcDriver</property>
    <property name="hibernate.connection.url">jdbc:hsqldb:hsql://localhost</property>
    <property name="hibernate.connection.username">sa</property>
    <property name="hibernate.dialect">org.hibernate.dialect.HSQLDialect</property>

    <property name="hibernate.hbm2ddl.auto">create</property>

    <!-- Use the C3P0 connection pool provider -->
    <property name="hibernate.c3p0.min_size">5</property>
    <property name="hibernate.c3p0.max_size">20</property>
    <property name="hibernate.c3p0.timeout">300</property>
    <property name="hibernate.c3p0.max_statements">50</property>
    <property name="hibernate.c3p0.idle_test_period">3000</property>

    <!-- Show and print nice SQL on stdout -->
    <!--
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
    -->

    <!-- List of annotated classes -->
    <mapping class="pos.logic.Store"/>
    <mapping class="pos.logic.Catalog"/>
    <mapping class="pos.logic.Product"/>
    <mapping class="pos.logic.Sale"/>
    <mapping class="pos.logic.Item"/>
    <mapping class="pos.logic.Payment"/>
    <mapping class="pos.logic.CashPayment"/>
    <mapping class="pos.logic.CheckPayment"/>

  </session-factory>
</hibernate-configuration>
```

Lista de todas las clases que contienen anotaciones de persistencia.

Notar que se pone el nombre completo de la clase (i.e. incluyendo el nombre del package que la contiene).

Igualmente, el archivo de configuración de Log4J (log4j.properties) debe estar localizado también en la raíz del classpath de Java. Si tal archivo no existe, crearlo copiando el del ejemplo del POS.