Práctica 7
Objetivo: Implementación de una capa de persistencia con MyBatis y Mybatis Generator

Se desea ampliar el ejemplo **cochelbatis** que se ofrece, incorporando la gestión de una nueva tabla 'Personas' y un procedimiento almacenado 'ventaCoche'.

Entre 'Personas' y 'Coche' existe una relación 1:N a través del atributo idPropietario de la tabla 'Coche'.

Se pide:

1- Ejecutar en SQLServer el script de creación de la base de datos proporcionado.

2- Ampliar el 'modelo' de clases del proyecto 'cochelbatis' para dar cabida a la nueva entidad 'Personas'.

3- Ampliar la capa de datos 'DAO' del proyecto para dar cabida a la nueva entidad con las operaciones habituales (insertar, modificar, borrar, buscar por la clave, por nif y obtener la lista de personas. Tener en cuenta que además la entidad 'Personas' debe recoger la información de la persona junto con los coches con los que está relacionado. Implementar un método que permita obtener la lista de coches de una persona concreta y otro que recupere la lista completa de las personas junto con sus coches.

4- Mejorar el método de inserción de un nuevo coche para que obtenga el valor de la primary key del objeto después de haber insertado el objeto en la B.D.

5- Implementar un método que permita hacer una llamada al procedimiento almacenado que cambia el propietario de un coche utilizando el framework de persistencia. El método debe retornar un valor que indique si se ha ejecutado de forma correcta o no.

6- Implementar un segundo método con la funcionalidad anterior utilizando transacciones de MyBatis.

7- Ampliar el programa principal del proyecto para probar todos los métodos.

8- Explicación sobre la instalación y uso de Mybatis Generator. La documentación y el software se pueden descargar de: http://code.google.com/p/mybatis/wiki/Generator
y el plug-in para Eclipse:
http://mybatis.googlecode.com/svn/sub-projects/generator/trunk/eclipse/UpdateSite/

La instalación de este plug-in se debe realizar a través de la opción para la instalación de nuevas funcionalidades de Eclipse.

9- A continuación, sustituir la capa de persistencia desarrollada por la generada por Mybatis Generator:

- Los .java correspondientes del modelo (coche y persona)
- La interfaz del cliente (DAO)
- Los map XML.

Y adaptar y extender estas salidas para implementar las mismas funcionalidades realizadas en los apartados anteriores.

Observaciones:

A) Para obtener el último valor de la columna identity del objeto insertado en la base de datos y poder mapearlo al objeto java que se ha insertado, hay que utilizar el siguiente elemento en el archivo XML de mapeo después de la inserción, donde id es el nombre la columna identidad en el objeto java.

```
<selectKey resultClass="int"  keyProperty="id" >
        select @@IDENTITY as value
 </selectKey>
```

Este elemento descriptivo, debe estar situado entre la etiqueta 'insert' y fin de 'insert' y después de la sentencia de inserción.

B) Para realizar una llamada a un procedimiento almacenado hay que seguir los siguientes pasos:

1- En el método que realiza la llamada:

a) Crear un objeto HashMaP
b) Añadir al objeto los pares formados por la etiqueta del parámetro y objeto con su valor.
c) Realizar la llamada
d) Recoger resultados (si el procedimiento retorna resultados)

Ejemplo:

```
HashMap parametros = new HashMap();

parametros.put("etiqueta1", new Integer(valor));
parametros.put("salida", new Integer(0));

sqlMap.update("sp", parametros);
Integer res = (Integer) parametros.get("salida");
```

El archivo XML que establece el mapeo debe tener la siguiente estructura:

```
<parameterMap id="spParameters" class="java.util.Map" >
     <parameter property="etiqueta1"
            javaType="java.lang.Integer" mode="IN"/>

<parameter property="salida"  jdbcType="INTEGER"
            javaType="java.lang.Integer" mode="INOUT"/>
  </parameterMap>

  <procedure id="sp"
    parameterMap="spParameters" >
    {call sp (?, ?, ?, ?)}
  </procedure>
```

C) Para cargar la información de un objeto y de todos los objetos relacionados en un solo paso es necesario diseñar una consulta, que obtenga todos los datos necesarios de las tablas de la BD que intervienen en la relación, y diseñar un resultmap con la estructura del objeto compuesto.

Ejemplo de una relación 1:N

```
<resultMap id="listaHijos" class="unican.taller.model.hijo">
```

```
    <result property="idHijo" column="id" />
    <result property="etiqHijo" column="columhijo1" />
</resultMap>

<resultMap id="padreResult" class="unican.taller.model.Padre" groupBy="idPadre" >
  <result property="idPadre" column="idPadre" />
  <result property="etiqPadre" column="columPadre1" />
  <result property="listaHijos"  column="idPadre"  resultMap="listaHijos"/>
</resultMap>

<select id="getPadre" resultMap="padreResult">
    SELECT
        p.IDPadre as idPadre,
        p.datoPadre as columnPadre1,
        h.IDHijo as id,
        h.datoHijo as columnhijo1
        FROM padre p left join hijo h on c.idPadre = h.idPersona
                WHERE p.IdPadre = #valor#
</select>
```