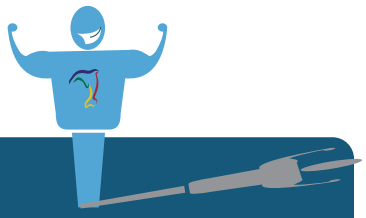
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

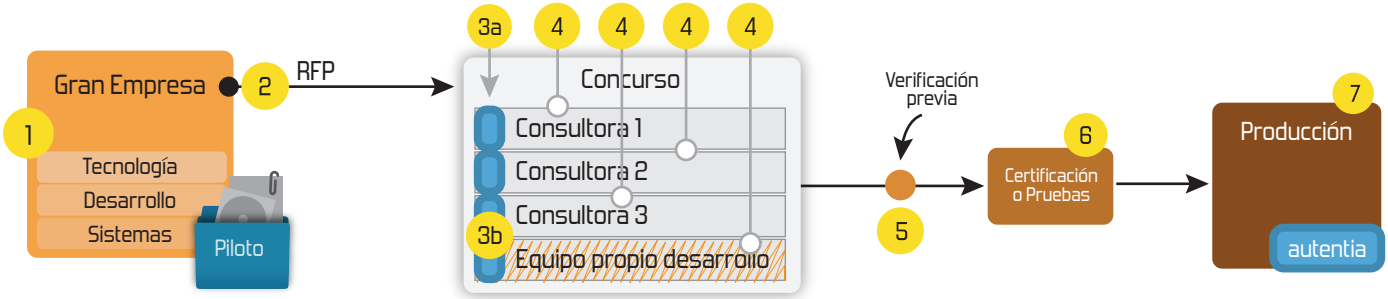
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por: Cesar Crespo Martín 2003.

1.Introducción

Trabajar con software orientado a objetos y bases de datos relacionales puede hacernos invertir mucho tiempo en los entornos actuales. Hibernate es una herramienta que realiza el *mapping* entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. El término utilizado es ORM (object/relational mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

Hibernate no solo realiza esta esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo.

2. Entorno

Las herramientas usadas en este tutorial son las siguientes:

- MySQL 4.1.1
- JDBC Driver: mysql-connector-java-3.0.11-stable-bin.jar
- MySQL Control Center 0.9.4
- Apache ANT 1.6
- Hibernate 2.1.4 instalado en el directorio C:\java\hibernate-2.1\

3. Conceptos básicos de Hibernate

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO, a veces llamado Plain Ordinary Java Object). Un POJO es similar a una Java Bean, con propiedades accesibles mediante métodos setter y getter, como por ejemplo:

```
package net.sf.hibernate.examples.quickstart;

public class Cat {

    private String id;
    private String name;
    private char sex;
    private float weight;

    public Cat() {
    }

    public String getId() {
        return id;
    }

    private void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public char getSex() {
        return sex;
    }
```

```
    public void setSex(char sex) {
        this.sex = sex;
    }

    public float getWeight() {
        return weight;
    }

    public void setWeight(float weight) {
        this.weight = weight;
    }

}
```

Para poder asociar el POJO a su tabla correspondiente en la base de datos, Hibernate usa los ficheros hbm.xml.

Para la clase Cat se usa el fichero Cat.hbm.xml para mapearlo con la base de datos. En este fichero se declaran las propiedades del POJO y sus correspondientes nombres de columna en la base de datos, asociación de tipos de datos, referencias, relaciones x a x con otras tablas etc:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
    PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

<hibernate-mapping>

    <class name="net.sf.hibernate.examples.quickstart.Cat" table="CAT">

        <!-- A 32 hex character is our surrogate key. It's automatically
            generated by Hibernate with the UUID pattern. -->
        <id name="id" type="string" unsaved-value="null" >
            <column name="CAT_ID" sql-type="char(32)" not-null="true"/>
            <generator class="uuid.hex"/>
        </id>

        <!-- A cat has to have a name, but it shouldn' be too long. -->
        <property name="name">
            <column name="NAME" length="16" not-null="true"/>
        </property>

        <property name="sex"/>

        <property name="weight"/>

    </class>

</hibernate-mapping>
```

De esta forma en nuestra aplicación podemos usar el siguiente código para comunicarnos con nuestra base de datos:

```
sessionFactory = new Configuration().configure().buildSessionFactory();
Session session = sessionFactory.openSession();


Transaction tx= session.beginTransaction();

Cat princess = new Cat();
princess.setName("Princess");
princess.setSex('F');
princess.setWeight(7.4f);

session.save(princess);
tx.commit();

session.close();
```

Además tiene la ventaja de que nos es totalmente transparente el uso de la base de datos pudiendo cambiar de base de datos sin necesidad de cambiar una línea de código de nuestra aplicación, simplemente cambiando los ficheros de configuración de Hibernate.

4. Confiración de Hibernate

Hasta ahora hemos obviado un dato importante en la configuración de Hibernate. Los datos de configuración de la base de datos

Podemos usar un fichero hibernate.properties o hibernate.cfg.xml que debe estar en el path de la aplicación:

hibernate.properties:

```
## MySQL
```

```
hibernate.dialect net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class org.gjt.mm.mysql.Driver
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql:///test
hibernate.connection.username cesar
hibernate.connection.password
```

hibernate.cfg.xml - Conexión mediante Datasource:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.datasource">
       java:comp/env/jdbc/shop</property>

    <property name="dialect">net.sf.hibernate.dialect.MySQLDialect
       </property>
    <property name="use_outer_join">true</property>
    <property name="transaction.factory_class">
       net.sf.hibernate.transaction.JDBCTransactionFactory</property>

    <property name="show_sql">true</property>

    <!-- Mapping files -->
    <mapping resource="com/shop/Category.hbm.xml" />
    <mapping resource="com/shop/Product.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

hibernate.cfg.xml - Conexión Directa:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="connection.driver_class">
com.mysql.jdbc.Driver</property>
<property name="connection.url">
jdbc:mysql://localhost/test</property>
<property name="connection.username">cesar</property>
<property name="connection.password"></property>

<property name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
<property name="use_outer_join">true</property>
<property name="transaction.factory_class">net.sf.hibernate.transaction.JDBCTransactionFactory</property>
<property name="show_sql">true</property>

<!-- Mapping files -->
    <mapping resource="com/shop/Category.hbm.xml" />
    <mapping resource="com/shop/Product.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

En caso de encontrarse ambos ficheros el .properties y el .hbm.xml se usara el .hbm.xml. Desde la aplicación también podemos especificar el fichero a usar:

```
SessionFactory sf = new Configuration()
   .configure("catdb.cfg.xml")
   .buildSessionFactory();
```

En estos ficheros se indican los parámetros de conexión de la base de datos como la base de datos a la que conectar, usuario y password etc.

Un parámetro interesante es el Dialecto de Hibernate. En este parámetro se indica el nombre de la clase que se encargará de comunicarse con la base de datos en el SQL que entienda la base de datos. Este parámetro ha de ser siempre especificado. El valor ha de ser una subclase que herede de net.sf.hibernate.dialect.Dialect

Hibernate nos proporciona los siguientes dialectos:

RDBMS	Dialect
DB2	net.sf.hibernate.dialect.DB2Dialect
MySQL	net.sf.hibernate.dialect.MySQLDialect
SAP DB	net.sf.hibernate.dialect.SAPDBDialect
Oracle (any version)	net.sf.hibernate.dialect.OracleDialect
Oracle 9	net.sf.hibernate.dialect.Oracle9Dialect
Sybase	net.sf.hibernate.dialect.SybaseDialect
Sybase Anywhere	net.sf.hibernate.dialect.SybaseAnywhereDialect
Progress	net.sf.hibernate.dialect.ProgressDialect
Mckoi SQL	net.sf.hibernate.dialect.MckoiDialect
Interbase	net.sf.hibernate.dialect.InterbaseDialect
Pointbase	net.sf.hibernate.dialect.PointbaseDialect
PostgreSQL	net.sf.hibernate.dialect.PostgreSQLDialect
HypersonicSQL	net.sf.hibernate.dialect.HSQLDialect
Microsoft SQL Server	net.sf.hibernate.dialect.SQLServerDialect
Ingres	net.sf.hibernate.dialect.IngresDialect
Informix	net.sf.hibernate.dialect.InformixDialect
FrontBase	net.sf.hibernate.dialect.FrontbaseDialect

Aquí observamos la gran importancia del fichero de configuración, pues es aquí donde se especifica que base de dato usamos, por lo que si cambiáramos de base de datos bastaría con cambiar este fichero de configuración, manteniendo nuestra aplicación intacta.

5. Hibernate Query Languaje HQL

Hibernate nos proporciona además un lenguaje con el que realizar consultas a la base de datos.

Este lenguaje es similar a SQL y es utilizado para obtener objetos de la base de datos según las condiciones especificadas en el HQL.

El uso de HQL nos permite usar un lenguaje intermedio que según la base de datos que usemos y el dialecto que especifiquemos será traducido al SQL dependiente de cada base de datos de forma automática y transparente.

Así una forma de recuperar datos de la base de datos con Hibernate sería:

Hibernate	JDBC

Session session = sessionFactory.openSession();
List cats = null;
try {
categories = session.find("from Cat");
 	
Iterator i = categories.iterator();
while (i.hasNext() == true) {
Cat cat = (Cat)i.next();
...
}
} finally {
session.close();
}
	Driver d =
	(Driver) Class.forName("com.mysql.jdbc.Driver").newInstance();
	DriverManager.registerDriver(d);
	
	try {
	Connection con = DriverManager.getConnection(
	"jdbc:mysql://yamcha/test",
	"cesar",
	"");
	
	Statement stmt = con.createStatement();
	
	String select =
	"SELECT * from cat";
	
	ResultSet res = stmt.executeQuery(select);
	
	while (res.next() == true) {
	String catID = res.getString("id");
	String catName = res.getString("name");
	
	Cat cat = new Cat(catID,catName);
	(.....)
	
	list.add(cat);

```
}
stmt.close();
con.commit();
con.close();

} catch (Throwable ex) {
System.out.println(" Error visualizando datos ");
}
```

De esta forma:

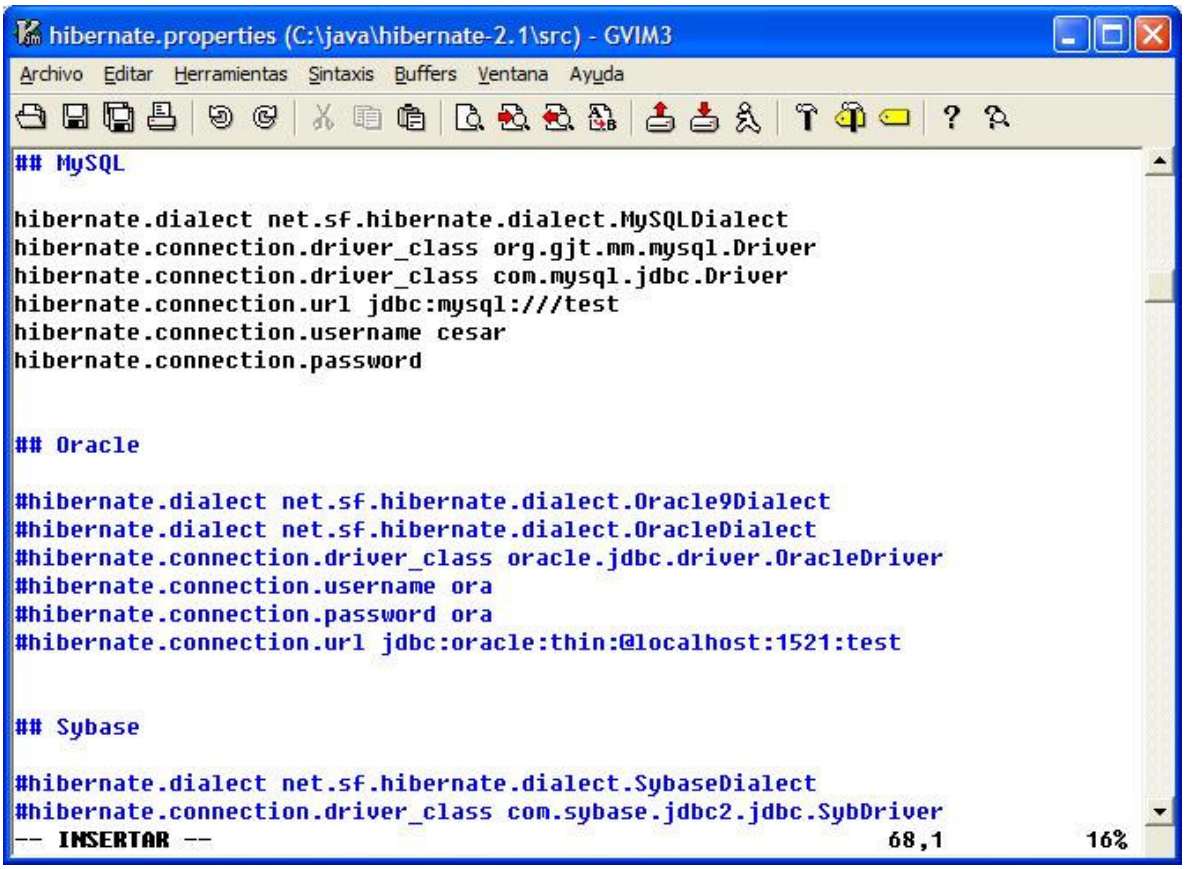
HQL	SQL
from Cat	select * from cat

Como podemos observar se simplifica considerablemente el código, así como se desacopla el uso de la base de datos de nuestra lógica de aplicación.

6. Ejempo de Hibernate

Una vez descargado y descomprimido Hibernate, en el directorio eg existe un ejemplo del funcionamiento de Hibernate.

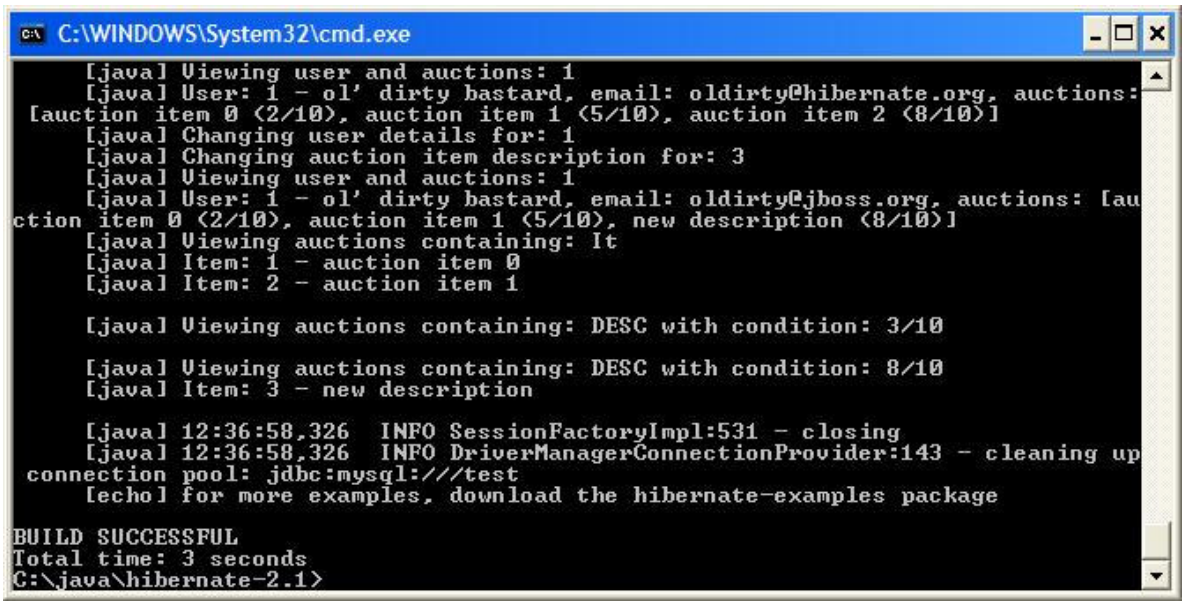
Puesto que nosotros usamos como base de datos MySQL debemos configurar adecuadamente el fichero C:\java\hibernate-2.1\src\hibernate.properties:



Además, debemos copiar el fichero mysql-connector-java-3.0.11-stable-bin.jar al directorio C:\java\hibernate-2.1\lib

Para ver el ejemplo debemos ejecutar en el directorio donde se encuentra Hibernate (C:\java\hibernate-2.1\): ant eg:

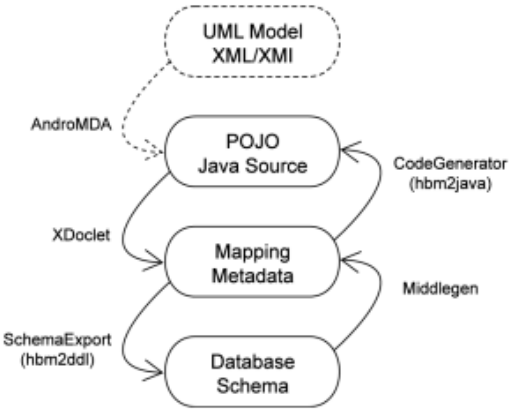
ANT nos compilara y ejecutará el ejemplo comprobando sus resultados por pantalla:



7. Herramientas usadas junto con Hibernate

Como se nos indica en la página de Hibernate http://www.hibernate.org/102.html

Existen diversas herramientas útiles para el uso de Hibernate que cubren todo el desarrollo desde nuestra aplicación hasta nuestra base de datos y viceversa:



Desde herramientas de modelado UML como por ejemplo con Poseidon podemos generar modelos entidad relación que son traducidos por AndroMDA a POJO's y mediante XDoclet se generan los ficheros HBM. Todas estas tareas se automatizan mediante el uso de ANT.

Otra opción es crear la base de datos con una herramienta de modelado y a partir de la base de datos una vez creada usar Middlegen para generar los ficheros HBM y a partir de estos los POJO's mediante hbm2java.

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
EJB´s y Orion	Recreación de la guía paso a paso de como crear una aplicación Web con EJB´s y Servlets y su despliegue con ANT sobre Orion
Otra implementación JDO con TJDO	Os mostramos como montar un ejemplo simple de JDO, a través de la implementación gratuita TJDO
Imágenes en Base datos y Java	Atendiento una de vuestras consultas, os mostramos como recuperar en Java, mediante JDBC una imagen almacenada en la base de datos MySQL
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
Reingeniería JDO con Druid	Os mostramos como crear vuestras clases y descriptores JDO, de tablas existentes, con la herramienta gratuita Druid.
Patrones de diseño J2EE	Os mostramos una interpretación particular de los patrones de diseño J2EE
Ado y Net	En este tutorial podeis descubrir como funciona ADO y las nuevas diferencias en arquitectura .Net .
CMP Entity Beans y MySql	Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql
Generación automática de código JDBC	En este tutorial os enseñamos como, sin conocimiento de JDBC, crear vuestro programas en Java, gracias a JDBCTest.

www.AdictosAlTrabajo.com Opimizado 800X600