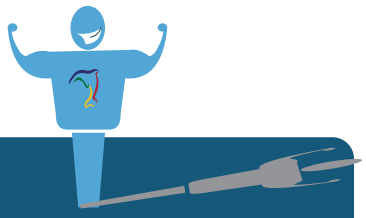
autentia

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...

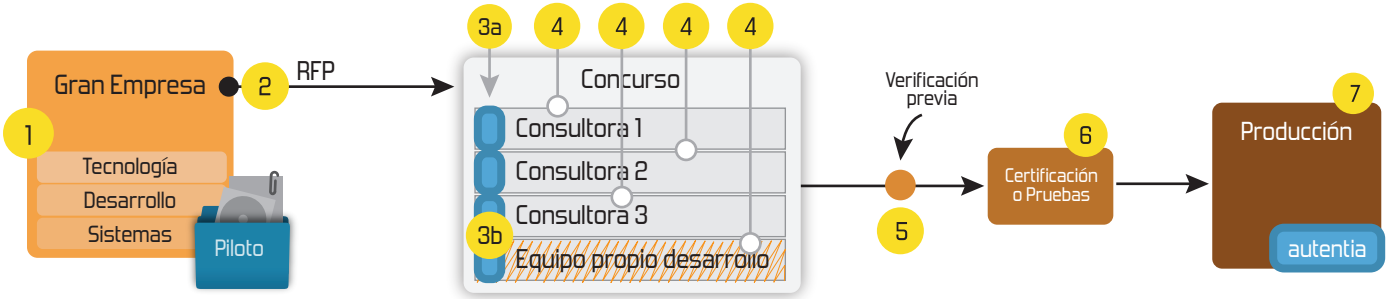
1. Desarrollo de componentes y proyectos a medida



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

Hosting Patrocinado por enREDados.com

Tutorial desarrollado por: Cesar Crespo Martín 2003.

Si te gusta lo que ves, **puedes contratarnos** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

Contacta: cesar@autentia.com.

autentia real business solutions

Descargar este documento en formato PDF middlegen.pdf

Modelado de bases de datos e integración con Hibernate mediante Middlegen

1. Introducción

Con este tutorial se pretende mostrar una guía de desarrollo de base de datos desde su modelado hasta su integración con Hibernate mediante Middlegen.

Middlegen es una potente herramienta que permite a partir de nuestra base de datos generar la capa de persistencia de EJB (CMP 2.0), JDO, Hibernate, Torque y además un sistema GUI de navegación de nuestra base de datos con JSP/Struts.

Esta herramienta puede ser descardada desde SourceForge: http://sourceforge.net/project/showfiles.php?group_id=36044

El uso de Middlegen se automatiza mediante el uso de Apache ANT.

Basándonos en este ejemplo podemos ahora nuestros POJO's (Plain Old Java Objects) y ficheros hbm.xml.

Con estos tres pasos cubrimos todo el desarrollo, desde el diseño de la base de datos hasta la generación de los ficheros descriptores de cada tabla de Hibernate y los POJO's para usar con Hibernate.

2. Entorno

Las herramientas usadas en este tutorial son las siguientes:

- DBDesigner 4.0.5.6
- MySQL 4.1.1, usaremos tablas de tipo InnoDB.
- mysql-connector-java-3.0.11-stable-bin.jar JDBC Driver
- MySQL Control Center 0.9.4
- Apache ANT 1.6
- Middlegen 2.0 instalado en el directorio C:\java\middlegen-2.0-vo
- Hibernate 2.1.4

3. Guía de desarrollo

El método de desarrollo de nuestra base de datos desde su diseño inicial hasta su uso en nuestra aplicación sería el siguiente:

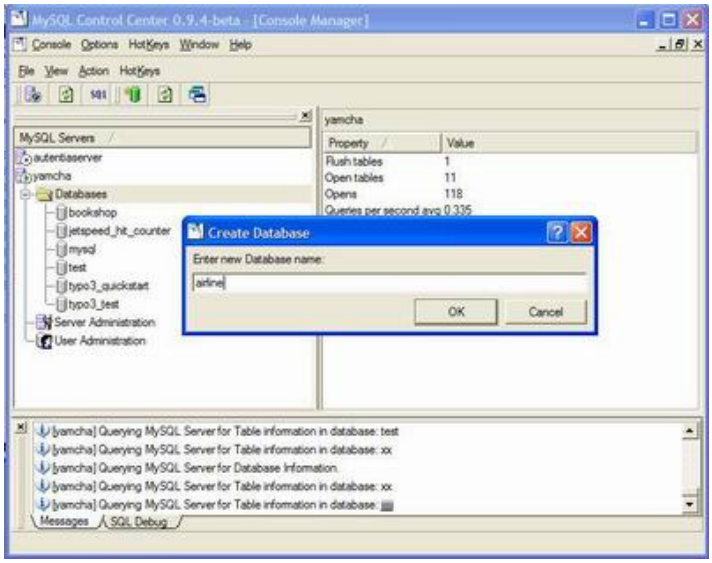
1. Modelado de la base de datos (Por ejemplo con DBDesigner).
2. Creación de la base de datos a partir de la herramienta de modelado.
3. Generación de los ficheros hbm.xml de Hibernate y los POJO's mediante Middlegen y hbm2java.

4. Ejemplo de Middlegen

Dentro del directorio C:\java\middlegen-2.0-vo\samples podemos encontrar un ejemplo donde se usa una base de datos de una

línea aérea.

Esta base de datos debe ser creada antes de ejecutar ant:



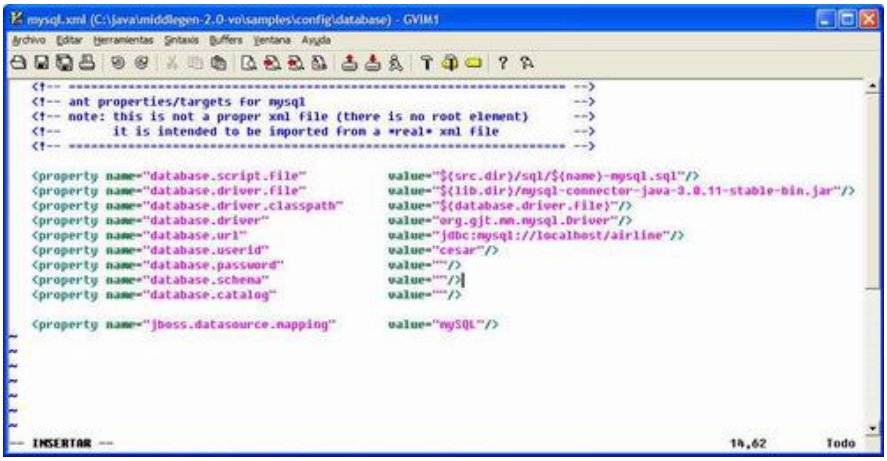
Middlegen es llamado desde ANT. Para que funcione este ejemplo debemos configurar los ficheros:

- build.properties.sample: Debemos renombrar este fichero como build.properties. En este fichero indicaremos los valores de las diferentes variables necesarias para el fichero build.xml. Aquí un parámetro interesante es *gui* que nos indica si usar la herramienta GUI de middlegen o no.
- build.xml : En este fichero especificamos que se debe incluir el fichero correspondiente a la base de datos que usemos. Puesto que usamos MySQL usamos el fichero ../config/database/mysql.xml el cual por defecto se conectará con la base de datos airlines que se encontrara en el localhost.

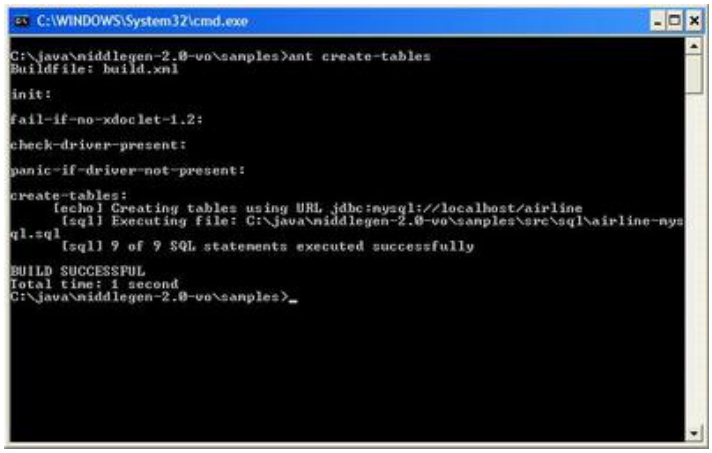
```
<!DOCTYPE project [
<!--<!ENTITY database SYSTEM "file:./config/database/hsqldb.xml">-->
    <!ENTITY database SYSTEM "file:./config/database/mysql.xml">

    <!ENTITY ejb      SYSTEM "file:./config/ejb/jboss.xml">
    <!--ENTITY ejb      SYSTEM "file:./config/ejb/weblogic.xml"-->
    <!--ENTITY ejb      SYSTEM "file:./config/ejb/mycsoft.xml"-->
]>
```

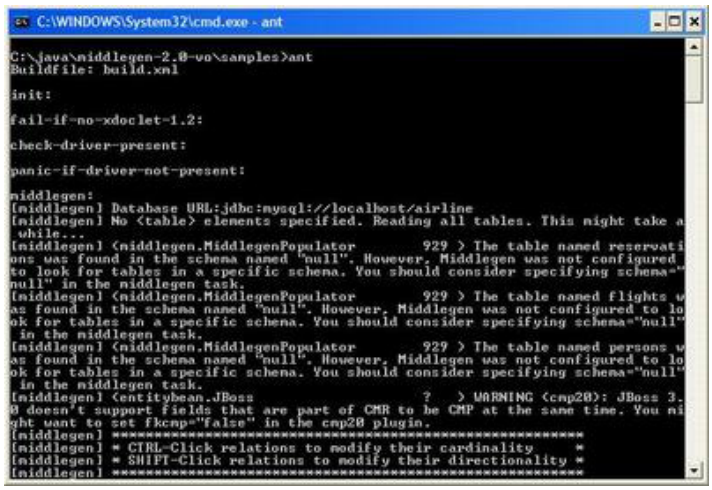
- mysql.xml: En este fichero debemos configurar el usuario y password así como el driver que usará para conectarse con la base de datos.
  Debemos añadir el driver mysql-connector-java-3.0.11-stable-bin.jar al directorio C:\java\middlegen-2.0-vo\samples\lib puesto que con el de por defecto el 3.0.0 no es capaz de conectar con nuestro MySQL 4.1.1.



Una vez configurados estos parámetros podremos crear la base de datos usando: ant create-tables:

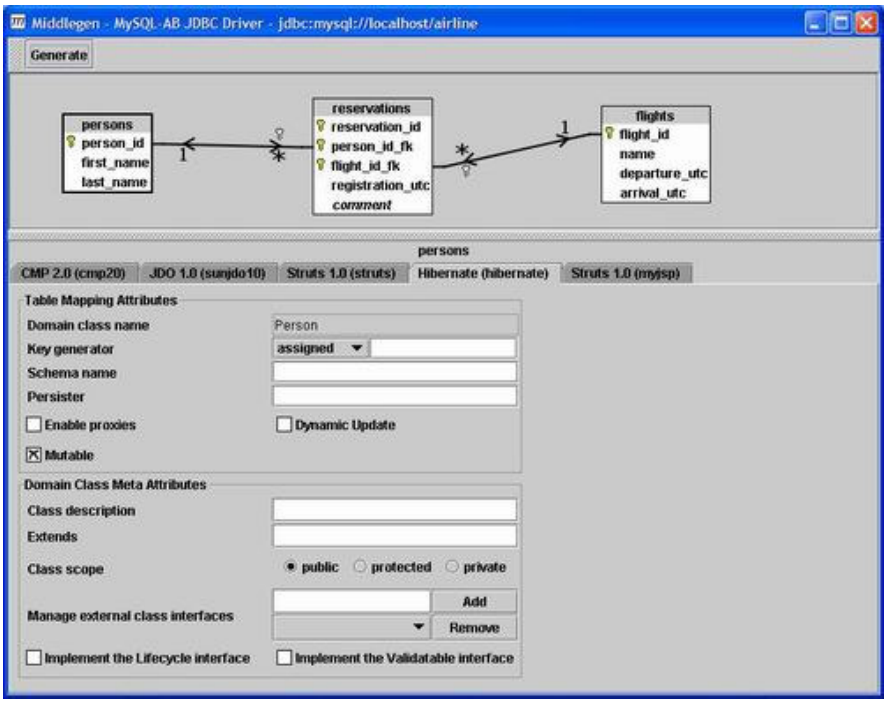


A continuación ejecutando ant obtendremos los siguientes resultados:



Puesto que en el fichero build.properties aparece gui=true, se nos aparecerá la herramienta visual de Hibernate, una vez aprendida la mecánica de Middlegen es recomendable saltarnos este paso para automatizar la tarea:

Esta herramienta visual muestra las tablas de la base de datos y las relaciones que existen entre ellas, pudiendo modificar estas relaciones así como distintos parámetros de JDO, Hibernate etc:



Pulsando el botón *Generate* se nos generarán los ficheros de descripción de Hibernate así como los Value Objects correspondientes a cada tabla.

Observamos que se crea el directorio build donde aparecerán todos los ficheros generados. De entre ellos destacamos:

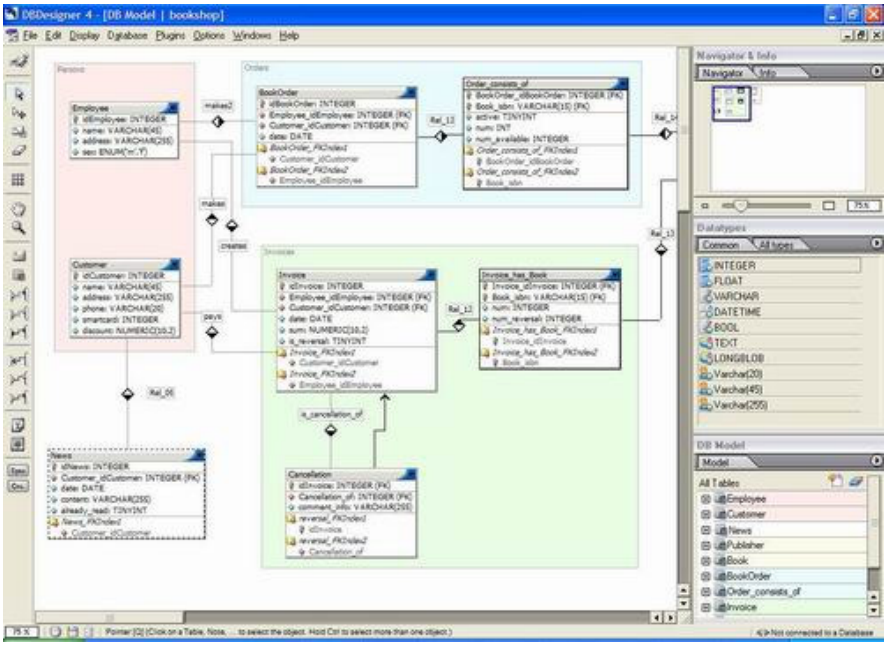
- C:\java\middlegen-2.0-vo\samples\build\gen-src\airline\hibernate : Directorio donde se generaran los ficheros hbm de Hibernate
- C:\java\middlegen-2.0-vo\samples\build\gen-src\airline\jdo : Ficheros Java.

También nos damos cuenta de que el GUI de Middlegen crea el fichero airline-prefs.properties en C:\java\middlegen-2.0-vo\samples\src guardando las relaciones y la posición de cada tabla dentro del diagrama. Mientras este fichero exista se mantendrán los cambios en las relaciones entre tablas aunque cambiemos la base de datos en el servidor.

5.Nuestro propio ejemplo

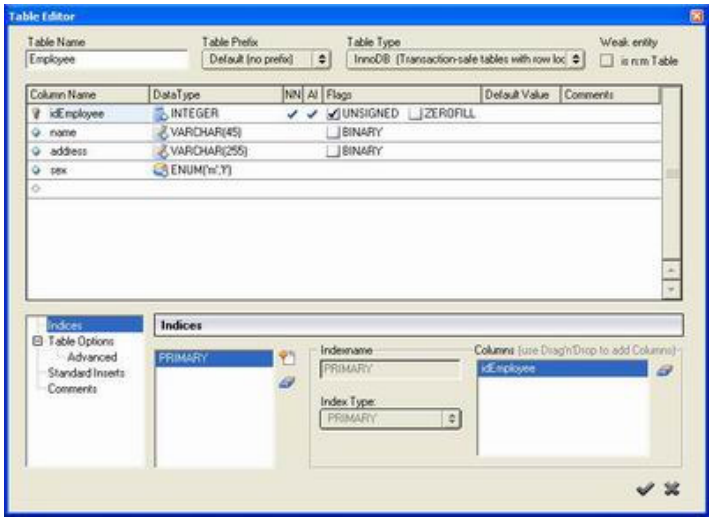
Siguiendo los pasos de la guía de desarrollo usaremos DBDesigner 4 para modelar nuestra base de datos MySQL.

Con DBDesigner podemos crear nuestra base de datos o importar un ejemplo del propio DBDesigner, así como hacer reingeniería sobre una base de datos ya creada. Si hemos instalado DBDesigner en C:\Program Files\fabFORCE\ usaremos el ejemplo C:\Program Files\fabFORCE\Examples\bookshop.xml:



Sobre este ejemplo realizaremos varios campos:

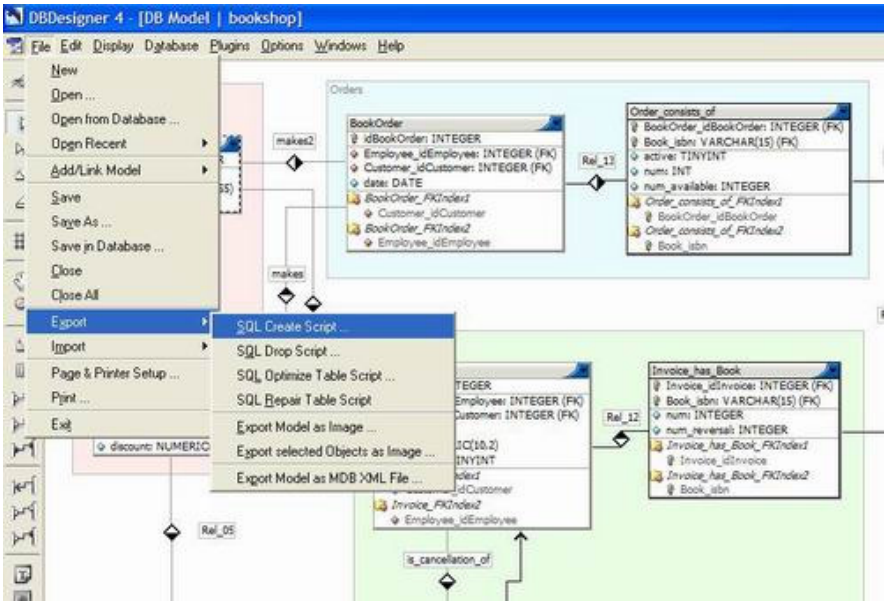
- En las propiedades de cada tabla seleccionaremos tipo InnoDB:
-



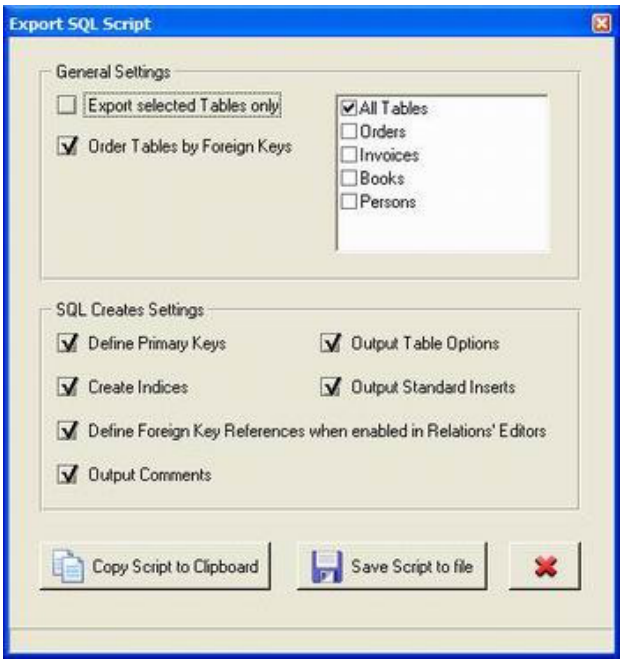
- Cambiaremos el tipo de datos BOOL por TINYINT con valores por defecto 0 en caso de ser FALSE y 1 en caso de ser

TRUE.

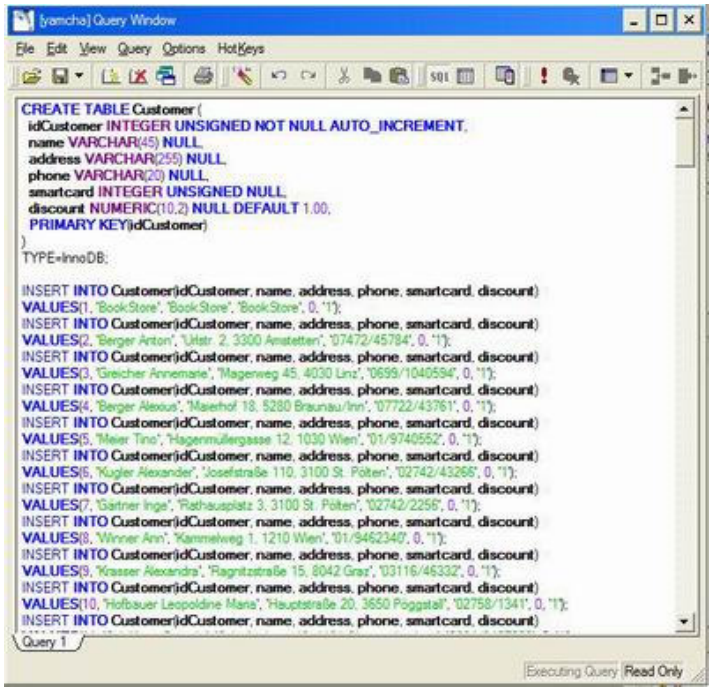
Una vez realizados estos cambios podremos exportar el diseño de la base de datos a DDL (Data Definition Language) obteniendo el SQL con las senticias CREATE de todas las tablas de nuestro diseño:



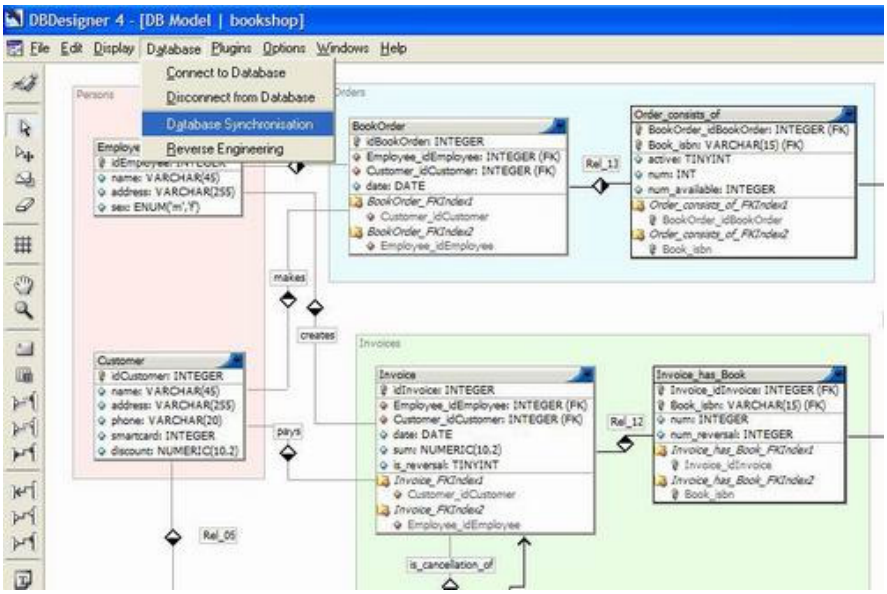
Podremos copiar estas sentencias SQL al porta papeles:

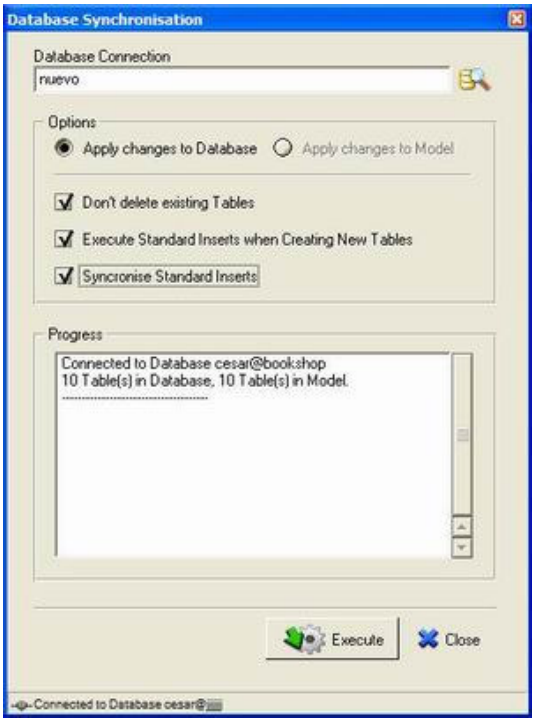


Para luego pegarlas en MySQL Control Center y crear nuestras tablas:

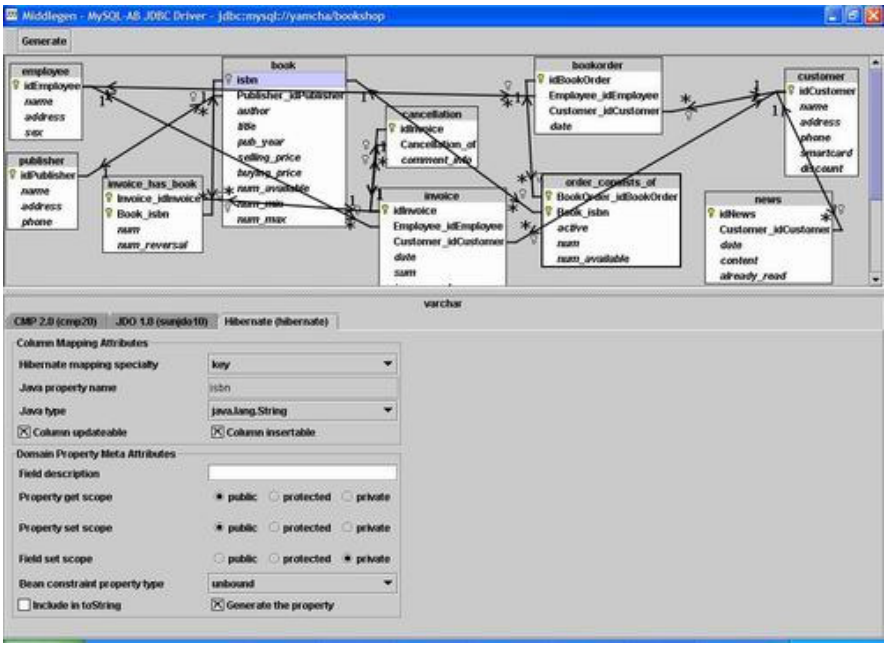


O bien usar la capacidad de sincronización de DBDesigner que nos creará automáticamente la base de datos y nos mantendrá sincronizados los cambios entre modelo y base de datos en ambos sentidos, es decir, si modificamos el modelo se actualizará la base de datos y viceversa, si modificamos la base de datos nos creará el modelo:

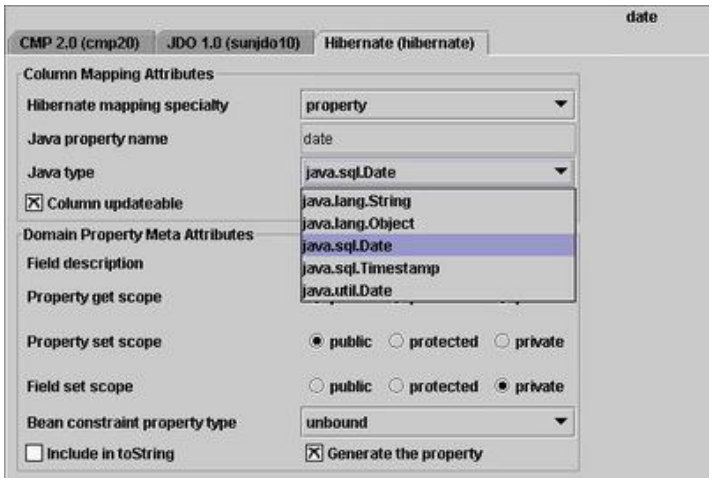




Una vez creada la base de datos a partir del modelo podemos usar Middlegen usando como fichero de partida el fichero build.xml de ejemplo. Nos bastará cambiar los parámetros de conexión de la base de datos del fichero mysql.xml indicándole ahora que la base de datos será bookshop:



Uno de los atractivos de la herramienta visual, es poder configurar tanto las relaciones entre tablas (poder cambiar las relaciones de 1..n a 1..n etc) y poder personalizar los tipos de datos por los que se traducirán los tipos de datos SQL a Java:



Un ejemplo útil es poder especificar sobre cada parámetro el tipo Java por el que se traducirá el tipo SQL, así para el tipo date podemos especificar el tipo Java por el que traducir.

6. Personalización de ANT

Dentro del fichero de ejemplo de ANT podemos encontrar elementos que no nos sean de utilidad por lo que podemos prescindir de ellos.

Los principales *target*s de nuestro fichero ant serán:

- Middlegen: Que nos generará los ficheros hbm.xml a partir de la base de datos:

```
<!-- ================================================================== -->
<!-- Clean everything                                                   -->
<!-- ================================================================== -->
<target name="clean" description="Clean all generated stuff">
   <delete dir="${build.dir}"/>
</target>

<!-- ================================================================== -->
<!-- Run Middlegen                                                      -->
<!-- ================================================================== -->
<target
   name="middlegen"
   description="Run Middlegen"
   unless="middlegen.skip"

>
   <mkdir dir="${build.gen-src.dir}"/>

   <taskdef
      name="middlegen"
      classname="middlegen.MiddlegenTask"
      classpathref="lib.class.path"
   />

   <middlegen
      appname="${name}"
      prefsdir="${src.dir}"
      gui="${gui}"
      databaseurl="${database.url}"
      initialContextFactory="${java.naming.factory.initial}"
      providerURL="${java.naming.provider.url}"
      datasourceJNDIName="${datasource.jndi.name}"
      driver="${database.driver}"
      username="${database.userid}"
      password="${database.password}"
      schema="${database.schema}"
      catalog="${database.catalog}"
   >
```

Este target es configurable en múltiples parámetros, así podemos indicarle que sólo genere determinadas tablas de nuestra base de datos:

```
<!--
We can specify what tables we want EJBs generated for.
If none are specified, EJBs will be generated for all tables.
Comment out the <table> elements if you want to generate for all tables.
Also note that table names are CASE SENSITIVE for certain databases,
so on e.g. Oracle you should specify table names in upper case.

The sequencename attribute is optional and will be used from
the oraclesequence extension. If no sequencename is present
it will be the name attribute suffixed by _SEQ.

-->
<!--table generate="true" name="persons" pktable="persons_pk"
        sequencename="my_persons_sequence" />
<table generate="true" name="flights" pktable="flights_pk"/>
<table name="reservations"/-->

<!--
If you want m:n relations, they must be specified like this.
Note that tables declare in multiple locations must all have
the same value of the generate attribute.
-->
<!--many2many>
   <tablea generate="true" name="persons"/>
   <jointable name="reservations" generate="false"/>
   <tableb generate="true" name="flights"/>
</many2many-->
```

- hbm2java : que nos generara los POJO's .java desde los ficheros hbm.xml:

```
<!-- ================================================================== -->
<!-- Run hbm2java                                                       -->
<!-- ================================================================== -->
<target name="hbm2java" depends="middlegen" description="Generate .java from .hbm files.">
   <taskdef
      name="hbm2java"
      classname="net.sf.hibernate.tool.hbm2java.Hbm2JavaTask"
      classpathref="lib.class.path"
   />

            <hbm2java output="${build.gen-src.dir}">
                  <fileset dir="${build.gen-src.dir}">
                        <include name="**/*.hbm.xml"/>
                  </fileset>
            </hbm2java>
</target>
```

Si desea contratar formación, consultoria o desarrollo de piezas a medida puede contactar con

Somos expertos en:
J2EE, C++, OOP, UML, Vignette, Creatividad ..
y muchas otras cosas

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.

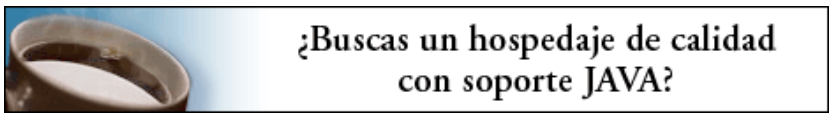
Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Comunicación entre TAGs, Beans y JSPs	Os mostramos las posibilidades de comunicación entre JSPs, Bean y etiquetas de usuario.
CMP Entity Beans y MySql	Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql
Introducción a Hibernate	Cesar Crespo nos enseña como utilizar unos de los sistemas más extendidos de mapeo

	de objetos a estructuras relacionales (tablas de base de datos)
Reingeniería JDO con Druid	Os mostramos como crear vuestras clases y descriptores JDO, de tablas existentes, con la herramienta gratuita Druid.
Otra implementación JDO con TJDO	Os mostramos como montar un ejemplo simple de JDO, a través de la implementación gratuita TJDO
Aplicación profesional con Struts	En este tutorial, os mostramos como crear una aplicación profesional, usando numerosos patrones y utilizando Struts.
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
Aplicación de Patrones de Diseño en Java	En este tutorial os mostramos como las técnicas avanzadas de diseño (como patrones de diseño) contribuyen a la contrucción de aplicaciones profesionales en Java.
Desarrollo Struts con XDoclet	Alejandro Perez nos enseña como simplificar el desarrollo de aplicaciones J2EE basadas en Struts, automatizando la generación de código con XDoclet
Modelado de MySQL con herramientas gratuitas	Os mostramos otra alternativa de modelado gráfico de MySql.

Patrocinados por enredados.com .... Hosting en Castellano con soporte Java/J2EE



www.AdictosAlTrabajo.com Opimizado 800X600