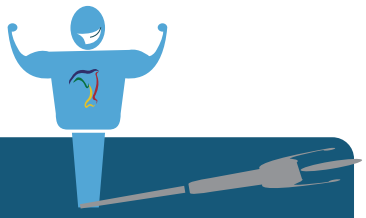
autentia

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...

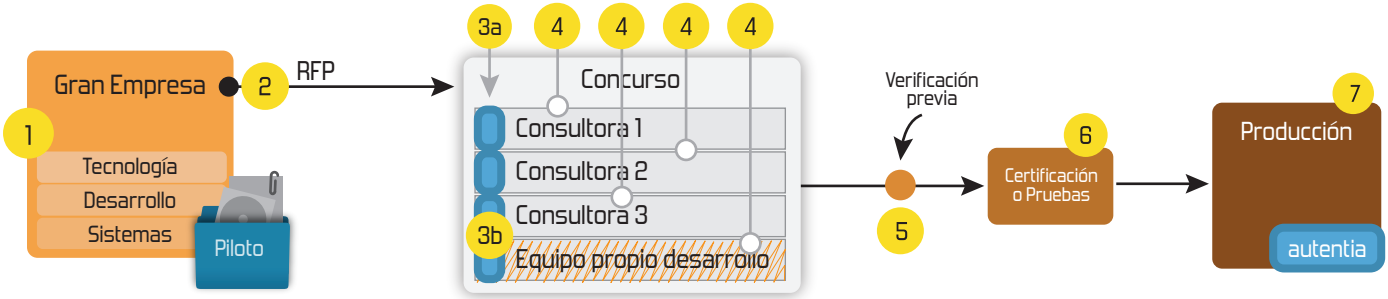
1. Desarrollo de componentes y proyectos a medida



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

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución

[¡NUEVO!] 2008-05-26 2008-05-19 2008-05-09 2008-04-20

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la seccion comic.

Tutorial desarrollado por

Daniel Hernandez del Peso

Consultor tecnológico de desarrollo de proyectos informáticos. Constructor de Adictos Al Trabajo 2.0

Ingeniero en Informática

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosAlTrabajo.com es el Web de difusión de conocimiento de Autentia.

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF: hbmFromAnnotations.pdf

Fecha de creación del tutorial: 2008-06-02

Ficheros de mapeo de Hibernate desde las clases

Introducción

En tutoriales anteriores hemos visto que, usando anotaciones, podemos crear clases que mapeen las entidades de una base de datos sin necesidad de ficheros de configuración (del tipo "miEntidad.hbm.xml").

Aunque esto es muy práctico, tiene el "inconveniente" (por ponerle pegas) de que las anotaciones sólo son válidas en entornos que empleen Java 5 o superiores... Por tanto puede darse el caso de que una aplicción que hemos diseñado con anotaciones tenga que ser migrada para adaptarla a un cliente que usa en su entorno Java 1.4 (y al que no podamos convencer para cambiar de versión de Java ;).

Esto supondría que tendríamos que eliminar del código todas las anotaciones y "picar" todo el código de los ficheros de mapeo de entidades.

En este tutorial vamos a intentar generar los ficheros de configuración de Hibernate de manera automática a partir de las clases anotadas, usando para ello las Hibernate Tools

Aspectos como la descarga e instalación de Hibernate Tools quedan fuera de este tutorial, pues hay otro (arriba tenéis el enlace) en el que se explica todo el proceso

Sí, sí, pero... ¿cómo se hace?

Una vez que hemos descargado las Hibernate Tools, tenemos dos opciones. Podemos usarlas como plugin de Eclipse o podemos usarlas mediante Ant

Como plugin de Eclipse

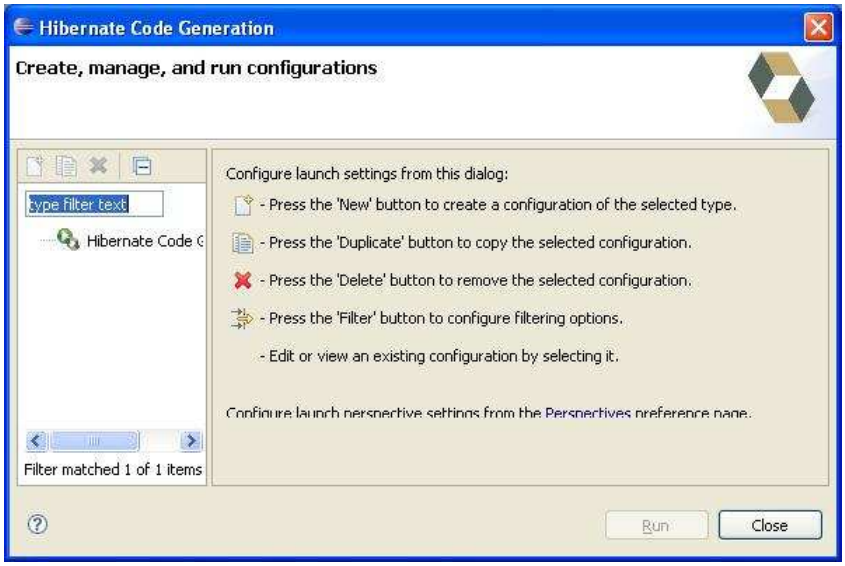
Una vez que tenemos las tools instaladas y la consola de configuración de Hibernate generada, vamos a usar dicha consola para hacer la magia...

Pinchamos en la flecha junto al botón adecuado de la barra de herramientas superior (el que está recuadrado)

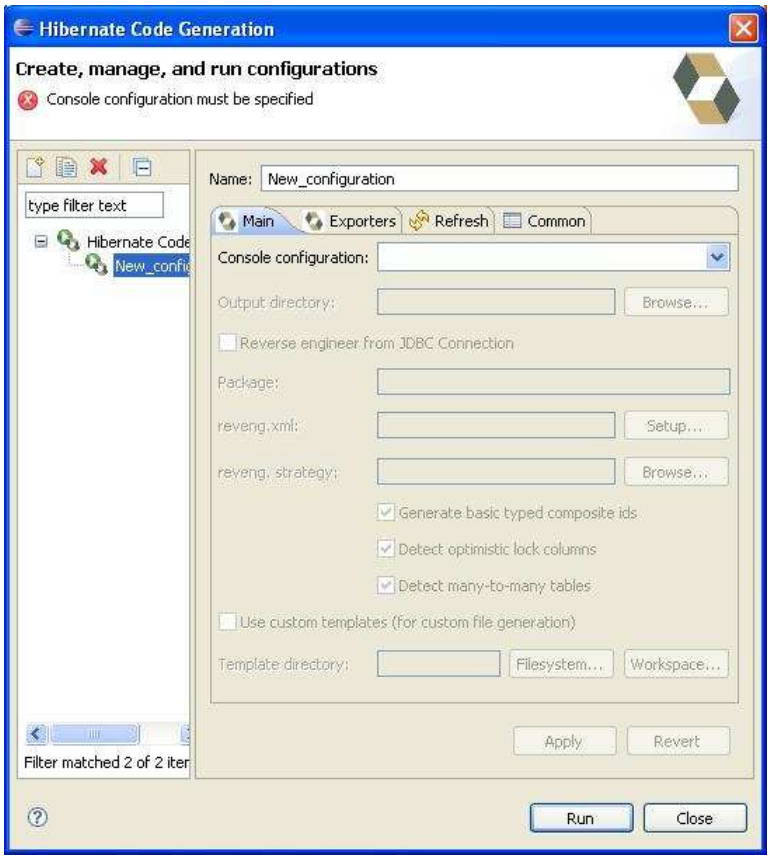
En el desplegable que aparece, seleccionamos "Open Hibernate Code Generation Dialog", y se nos abre la ventana siguiente:

Últimos tutoriales

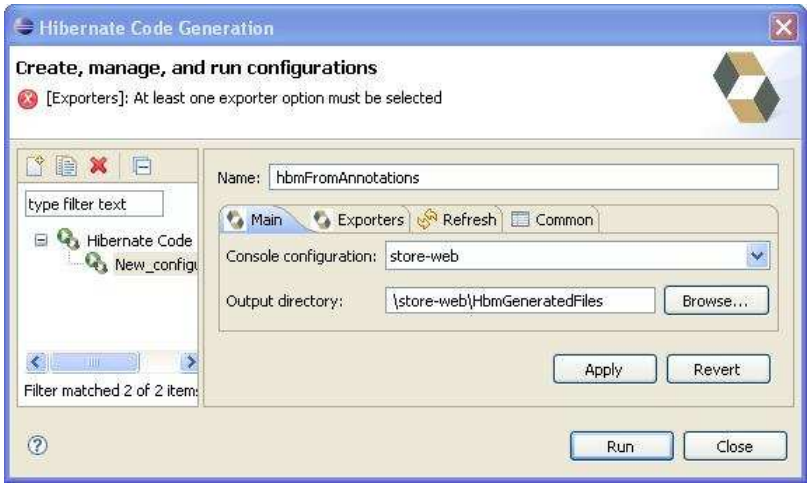
Últimas ofertas de empleo



Pulsamos el botón de "Nueva configuración":



En la ventana que aparece (pestaña "Main"), rellenamos el nombre de la nueva configuración, la configuración de consola que vamos a usar y el directorio en el que se almacenarán los ficheros generados



En la pestaña "Exporters" marcamos la opción "Hibernate XML Mappings" y pulsamos el botón "Run"... y la magia está hecha

Abrimos uno de los ficheros generados:

```
view plain print ?
01. <?xml version="1.0"?>
02. <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
03. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
04. <!-- Generated 16-may-2008 12:45:47 by Hibernate Tools 3.2.1.GA -->
05. <hibernate-mapping>
06.     <class name="com.autentia.store.User" table="User">
07.         <id name="id" type="java.lang.Integer" access="field">
08.             <column name="id" />
09.             <generator class="native"></generator>
10.         </id>
11.         <property name="address" type="java.lang.String" access="field">
12.             <column name="address" />
13.         </property>
14.         <property name="creditCard" type="java.lang.String" access="field">
15.             <column name="creditCard" />
16.         </property>
17.         <property name="email" type="java.lang.String" access="field">
18.             <column name="email" />
19.         </property>
20.         <property name="name" type="java.lang.String" access="field">
21.             <column name="name" />
22.         </property>
23.         <property name="password" type="java.lang.String" access="field">
24.             <column name="password" />
25.         </property>
26.     </class>
27. </hibernate-mapping>
```

Usando Ant

Ya hemos visto que, usando el plugin de Eclipse, la cosa es bastante sencilla... Ahora vamos a ver cómo se podría hacer lo mismo con Ant. Esta manera de resolver el problema nos da la ventaja de no necesitar Eclipse... un fichero BAT o un shell script podría lanzar automáticamente el proceso de creación.

Lo primero que necesitamos es un fichero ejecutable de ant, que llamaremos "build.xml"

En él, hay que definir la tarea de Ant para Hibernate Tools (hibernatetool). Además, hay que indicarle el classpath donde poder encontrar la clase que implementa la tarea. También vamos a necesitar las librerías tanto de Hibernate como de Hibernate Annotations, por lo que las pondremos ya en el classpath:

```
view plain print ?
01. <project basedir="."  >
02.     <property name="my.classpath" value="antClasspath"/>
03.     <path id="toolslib">
04.      <path location="${my.classpath}/hibernate-tools.jar"  />
05.      <path location="antClasspath/hibernate-3.2.6.ga.jar"  />
06.      <path location="antClasspath/hibernate-commons-annotations-3.3.0.ga.jar" />
07.      <path location="antClasspath/hibernate-annotations-3.3.0.ga.jar" />
08.     </path>
09.
10.     <taskdef name="hibernatetool"
11.              classname="org.hibernate.tool.ant.HibernateToolTask"
12.              classpathref="toolslib" />
13.
14. </project>
```

Y ahora hay que meter la llamada a la tarea Ant propiamente dicha. La tarea precisa un sistema de trazas (commons-login) y otras librerías que añadiremos al classpath (el resto de librerías necesarias las averiguamos por prueba/error; vemos lo que dice que falta y "googleamos" :-)

Otra cosa importante es que, cuando a la tarea le pasamos la ruta a las clases del proyecto, debemos pasarle la ruta a las clases *compiladas*

```
view plain print ?
01. <project basedir="." >
02.     <property name="my.classpath" value="antClasspath"/>
03.     <property name="build.dir" value="antHbmGeneratedFiles" />
04.
05.     <path id="toolslib">
06.      <path location="${my.classpath}/hibernate-tools.jar" />
07.      <path location="${my.classpath}/hibernate-3.2.6.ga.jar" />
08.      <path location="${my.classpath}/hibernate-commons-annotations-3.3.0.ga.jar" />
09.      <path location="${my.classpath}/hibernate-annotations-3.3.0.ga.jar" />
10.      <path location="${my.classpath}/commons-logging-1.1.1.jar" />
11.      <path location="${my.classpath}/commons-collections-3.2.1.jar" />
12.      <path location="${my.classpath}/dom4j-1.6.1.jar" />
13.      <path location="${my.classpath}/persistence-api-1.0.jar" />
14.      <path location="${my.classpath}/freemarker.jar" />
15.      <path location="${my.classpath}/Tidy.jar" />
16.     </path>
17.
18.     <taskdef name="hibernatetool"
19.              classname="org.hibernate.tool.ant.HibernateToolTask"
20.              classpathref="toolslib" />
21.
22.     <target name="init">
23.         <hibernatetool destdir="./${build.dir}">
24.              <classpath>
25.
26.                <path location="./target/classes" />
27.               </classpath>
28.             <annotationconfiguration configurationfile="src/main/resources/hibernate.cfg.xml" />
29.             <hbm2hbmxml/>
30.         </hibernatetool>
31.     </target>
32.
33. </project>
```

Os adjunto enlaces a las librerías:

- Librerías commons-*: página oficial de Apache Commons
- Librerías de Hibernate: página oficial de Hibernate. Commons Annotations es una librería que viene en el Zip de hibernate-annotations
- Dom4j: página oficial
- Freemarker: página de Freemarker en Source Forge
- Tidy: página de JTidy en Source Forge
- Persistence Api contiene el API de JPA, o sea que podéis encontrarlo en diversas fuentes

En cualquier caso, si utilizáis Maven en vuestros proyectos, la mayoría de estas librerías las podéis encontrar en los repositorios públicos.

Ejecutamos el Ant, y...

```
view plain print ?
01. <?xml version="1.0"?>
02. <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
03. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
04. <!-- Generated 16-may-2008 13:31:59 by Hibernate Tools 3.2.1.GA -->
05. <hibernate-mapping>
06.     <class name="com.autentia.store.User" table="User">
07.         <id name="id" type="java.lang.Integer" access="field">
08.             <column name="id" />
09.
10.             <generator class="native">
11.             </generator>
12.         </id>
13.
14.         <property name="address" type="java.lang.String" access="field">
15.             <column name="address" />
16.         </property>
17.
18.         <property name="creditCard" type="java.lang.String" access="field">
19.             <column name="creditCard" />
20.         </property>
21.
22.         <property name="email" type="java.lang.String" access="field">
23.             <column name="email" />
24.         </property>
25.
26.         <property name="name" type="java.lang.String" access="field">
27.             <column name="name" />
28.         </property>
29.
30.         <property name="password" type="java.lang.String" access="field">
31.             <column name="password" />
32.         </property>
33.     </class>
34. </hibernate-mapping>
```

Como vemos, los ficheros generados de ambas maneras son iguales, salvo por algún salto de línea. Ahora solo falta comprobar que los ficheros son correctos... Para ello, vamos a elminar las anotaciones de las clases (aunque no las quitemos, debería hacer caso a los ficheros antes que a las anotaciones, pero para estar seguros, las quitamos). Además de quitar las anotaciones, en el "hibernate.cfg.xml" cambiamos todas las entradas de tipo "<mapping class="..." />" por entradas del tipo "<mapping resource="mificherogenerado.hbm.xml" />".

Probando los ficheros

Lo primero que vemos es que, si tenemos clases anotadas que hereden de otras, obtenemos un error al intentar ejecutar nuestro proyecto... ya que, si bien en las subclases sí que genera el atributo "discriminator-value", en la clase base no genera la etiqueta "<discrminator column="" type="">". Veamos un ejemplo.

```
view plain print ?
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
<!-- Generated 19-may-2008 9:32:08 by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
    <class name="com.autentia.store.product.Product"  table="Product">
        <id name="id" type="java.lang.Integer">
            <column name="id" />

            <generator class="native">
            </generator>
        </id>

        <property name="description" type="java.lang.String">
            <column name="description" />
        </property>

        <property name="name" type="java.lang.String">
            <column name="name" />
        </property>

        <property name="price" type="float">
            <column name="price" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

Debería tener este otro aspecto:

```
view plain print ?
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
<!-- Generated 19-may-2008 9:32:08 by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
    <class name="com.autentia.store.product.Product"  table="Product">
        <id name="id" type="java.lang.Integer">
            <column name="id" />

            <generator class="native">
            </generator>
        </id>
<discriminator column="Type" type="string"/>
        <property name="description" type="java.lang.String">
            <column name="description" />
        </property>

        <property name="name" type="java.lang.String">
            <column name="name" />
        </property>

        <property name="price" type="float">
            <column name="price" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

Por otro lado, hay que tener en cuenta que nuestras clases tienen que tener los getters y setters para que se pueda acceder a los campos de la entidad

Otro fallo que he encontrado al generar los ficheros de mapeo automáticamente es que no convierte las anotaciones "@cascade". Por ejemplo, el siguiente fragmento de código

```
view plain print ?
@OneToMany(cascade = CascadeType.ALL)
private List<BuycartProduct> buycartProducts =  new ArrayList<BuycartProduct>();
```

Lo traduce de la siguiente manera:

```
view plain print ?
<!-- Todo el XML anterior -->
<bag name="buycartProducts" inverse="false">
    <key>
        <column name="Buycart_id" not-null="true" />
    </key>

    <many-to-many entity-name="com.autentia.store.BuycartProduct" >
        <column name="buycartProducts_id" not-null="true" />
    </many-to-many>
</bag>
<!-- Todo el XML posterior -->
```

Como podéis ver, la información sobre las acciones que deben ejecutarse en cascada se ha perdido... Esto causa problemas a la hora de persistir el objeto si no se persiste también explícitamente la entidad asociada ("BuycartProduct" en este caso). Lo correcto habría sido

```
view plain  print  ?
<!-- Todo el XML anterior -->
<bag name="buycartProducts" inverse="false" cascade="all">
    <key>
        <column name="Buycart_id" not-null="true" />
    </key>

    <many-to-many entity-name="com.autentia.store.BuycartProduct" >
        <column name="buycartProducts_id" not-null="true" />
    </many-to-many>
</bag>
<!-- Todo el XML posterior -->
```

Una vez que hemos tomado estas precauciones, empaquetamos nuestra aplicación y la desplegamos en nuestro servidor de aplicaciones (en mi caso, un Tomcat 6.0.16) y...

Conclusiones

Como podéis ver, es fácil transformar nuestro entorno de Hibernate basado en clases anotadas para usar ficheros de configuración XML, usando las Hibernate Tools... No obstante, si lo que queréis es migrar una aplicación con anotaciones a un entorno que utilice Java 1.4, recordad que no sólo hay que quitar las anotaciones, sino que hay que eliminar también la utilización de Generics (del tipo "List<String>"). Lo demás, como habéis visto, es sencillo...

Pero si a pesar de esta sencillez preferís que otros lo hagan por vosotros, os recuerdo que podéis contratar a Autentia, expertos en el desarrollo de aplicaciones usando las tecnologías más avanzadas (Spring, Hibernate, JSF, ICEFaces...)

- Puedes opinar sobre este tutorial haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati. ADD THIS BLOG TO MY Technorati FAVORITES

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subcripción a novedades:

E-mail [ ] Aceptar

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	pdf
Manejar dos bases de datos distintas con Hibernate	Alejandro Pérez nos enseña como manejar dos bases de datos distintas con Hibernate	2005-07-13	15451	pdf
Comparativa entre Hibernate y EJB3 en la Capa de Persistencia	El presente documento pretende dar algunas luces a la comparativa entre la opción de usar Hibernate y/ó EJB3 para la capa de persistencia	2007-08-16	5339	pdf
Hibernate 3 y los tipos de datos para cadenas largas	En este tutorial se contará la experiencia que hemos tenido a la hora de manejar los diferentes tipos de datos existentes para grandes cadenas de texto, tales como el tipo Clob de Oracle, o los tipos TEXT de MySQL y SQLServer, utilizando la última versión	2007-03-23	7554	pdf
Hibernate y el mapeo de la herencia	En este tercer tutorial de la saga vamos a ver en este tutorial como implementar las relaciones de herencia con las anotaciones de JPA	2007-06-27	3869	pdf
Hibernate Tools y la generación de código	En este tutorial vamos a ver como usar estas herramientas para hacer el esqueleto de una pequeña aplicación, de manera muy sencilla, generando código a partir de las tablas creadas en la base de datos.	2007-06-13	8686	pdf
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)	2004-08-14	53928	pdf
Creación automática de recursos Hibernate con Middlegen	En este tutorial aprendereis como utilizar la herramienta middlegen para generar distintas capas de persistencia (CMP 2.0, JDO, Hibernate, Torque), a partir de un modelo físico de datos, de un modo automático, mediante el uso de la herramienta middlegen	2004-08-26	25730	pdf
Hibernate 3.1, Colecciones, Fetch y Lazy	En este tutorial vamos a ver cómo se comportan ciertas relaciones, y cómo podemos optimizar las consultas a la base de datos con Hibernate	2006-04-17	15609	pdf
Hibernate y las anotaciones de EJB 3.0	En este tutorial Alejandro Pérez nos muestra las ventajas que nos aporta Hibernate y las anotaciones de EJB 3.0	2007-06-25	5486	pdf
Creación de una aplicación con Spring e Hibernate desde 0	Este tutorial vamos a explicar paso a paso cómo crear una pequeña aplicación usando Spring e Hibernate con anotaciones partiendo desde 0	2008-02-15	4642	pdf

Nota: