

¿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

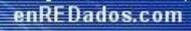
Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

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

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

		Hosting Patrocinado por  
---	---	---

[Home](#) | [Quienes Somos](#) | [Empleo](#) | [Tutoriales](#) | [Contacte](#)

<p>Tutorial desarrollado por: Alejandro Perez García 2003-2005, nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.</p> <p>Si te gusta lo que ves, puedes contratarme para impartir cursos presenciales en tu empresa o para ayudarte en proyectos (Madrid).</p> <p>Contacta: alejandropg@autentia.com.</p>	
---	---

Descargar este documento en formato PDF [hibernatec.pdf](#)

[Firma en nuestro libro de Visitas](#)

[Eclipse Hibernate Tools](#)

Java Data-object Hibernate
mapping Comprehensive J2EE IDE
& Support

[Formación Empresas](#)

Consultoría de Formación
Tecnologías Web

[Cursos J2ee](#)

La guía con la información que
buscas para ampliar tu formación.

[Web.XML- Java Config File](#)

Edit/Validate web.xml for J2EE
Apps Syntax Help, Easy-to-use,
Free D/L.

Anuncios Goooooogle

Anunciarse en este sitio

Como manejar dos bases de datos distintas con Hibernate

Índice de contenidos

1. Introducción
2. Entorno
3. Hibernate con más de una base de datos
4. Probando la clase de ayuda
5. Conclusiones

1. Introducción

En ciertas ocasiones nos vemos obligados a manejar en una aplicación más de una base de datos diferente.

En este tutorial vamos a exponer una clase de ayuda para poder gestionar esta situación con Hibernate.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Ahtex Signal X-9500M (Centrino 1.6 GHz, 1024 MB RAM, 60 GB HD).
- Sistema Operativo: GNU / Linux, Debian Sid (unstable), Kernel 2.6.11, KDE 3.4
- Máquina Virtual Java: JDK 1.5.0_03 de Sun Microsystems
- Hibernate 3.0 (<http://www.hibernate.org>)

3. Hibernate con más de una base de datos

En principio no hay ningún problema para poder gestionar mas de una base de datos con Hibernate.

En Hibernate podemos tener mas de un fichero de configuración, siendo el fichero de configuración por defecto "hibernate.cfg.xml". Cada fichero de configuración puede representar una conexión a una base de datos distinta, y por supuesto cada fichero de configuración hará referencia a los ficheros de mapeo correspondientes para las tablas de cada base de datos.

La clase que vamos a presentar a continuación pretende facilitar la gestión de esta situación, consiguiendo de forma sencilla sesiones de Hibernate a cada una de las bases de datos.

Además la clase se encarga de asegurar que la sesión de Hibernate es única por Thread. Esto puede resultar muy conveniente en aplicaciones multithread o en aplicaciones Web donde múltiples clientes acceden de forma simultánea.

El código de la clase es el siguiente:

```
package com.autentia.tool;

import java.util.HashMap;
import java.util.Map;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

/**
 * Clase de utilidad para conseguir la sesión de Hibernate.
 * La sesión es única por thread.
 * Pueden existir varios ficheros de configuración.
 * Al menos siempre existirá la configuración por defecto sacada del fichero "hibernate.cfg.xml".
 *
 * @author Autentia
 */
public class HibernateMultipleConfig {

    /** Clave para acceder a la sesión por defecto, esta sesión se obtiene a partir del fichero de configuración "hibernate.cfg.xml". */
    public static final String DEFAULT = "default";

    /** Mapa donde se guardan los SessionFactory para cada uno de los ficheros de configuración. */
    private static final Map<String, SessionFactory> sessionFactories = new HashMap();

    /** Atributo único por thread donde se guardan las sesiones para cada uno de los ficheros de configuración. */
    private static final ThreadLocal<Map<String, Session>> sessions = new ThreadLocal();

    static {
        try {
            // Create the DEFAULT SessionFactory from hibernate.cfg.xml
            SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
            sessionFactories.put(DEFAULT, sessionFactory);
        } catch (Throwable e) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + e);
            throw new ExceptionInInitializerError(e);
        }

        // Creamos el mapa donde se guardará la sesión para cada una de las configuraciones.
        // Este mapa lo guardamos en el atributo sessions, que nos asegura que el mapa es único por thread.
        sessions.set(new HashMap<String, Session>());
    }

    /**
     * Añade un nuevo fichero de configuración, creando el SessionFactory correspondiente.
     * Para encontrar el fichero de configuración, este debe encontrarse en el classpath,
     * junto al fichero de configuración por defecto de hibernate "hibernate.cfg.xml".
     *
     * @param fileName nombre del fichero de configuración que se quiere añadir.
     * @param key clave con la que se hará referencia a esta configuración en las siguientes llamadas.
     * @throws HibernateException si no se puede crear el SessionFactory basado en el fichero que se pasa como
     * argumento.
     */
    public static void addConfigFile(String fileName, String key) throws HibernateException {
        SessionFactory sessionFactory = new Configuration().configure(fileName).buildSessionFactory();
        sessionFactories.put(key, sessionFactory);
    }

    /**
     * Devuelve la sesión, para el thread que hace la llamada. Es decir cada thread tiene su propia sesión.
     *
     * @param key clave que identifica la sesión a la que se quiere acceder.
     * @return sesión por defecto, para el thread que hace la llamada.
     * @throws HibernateException
     */
    public static Session currentSession(String key) throws HibernateException {
        Map<String, Session> ss = sessions.get();
        Session s = ss.get(key);

        // Abre una nueva sesión si este thread todavía no tiene ninguna
        if (s == null) {
            s = sessionFactories.get(key).openSession();
            ss.put(key, s);
        }
        return s;
    }

    /**
     * Cierra la sesión, para el thread que hace la llamada.
     *
     * @param key clave que identifica la sesión que se quiere cerrar.
     * @throws HibernateException
     */
}
```

```

*/
public static void closeSession(String key) throws HibernateException {
    Map<String, Session> ss = sessions.get();
    Session s = ss.get(key);
    if (s != null) {
        s.close();
        ss.put(key, null);
    }
}

/**
 * Devuelve la sesión por defecto, para el thread que hace la llamada.
 * Es decir cada thread tiene su propia sesión.
 *
 * @return sesión por defecto, para el thread que hace la llamada.
 * @throws HibernateException
 */
public static Session currentSession() throws HibernateException {
    return currentSession(DEFAULT);
}

/**
 * Cierra la sesión por defecto, para el thread que hace la llamada.
 *
 * @throws HibernateException
 */
public static void closeSession() throws HibernateException {
    closeSession(DEFAULT);
}
}

```

Este código está basado en la clase de ayuda propuesta por la documentación de Hibernate en: http://www.hibernate.org/hib_docs/v3/reference/en/html/tutorial.html#tutorial-firstapp-helpers

4. Probando la clase de ayuda

Aquí proponemos un pequeño ejemplo para probar la clase de ayuda:

```

import org.hibernate.Session;

import com.autentia.tool.HibernateMultipleConfig;

public class TestHibernateMultipleConfig {

    public static void main(String[] args) {
        // Añadimos un nuevo fichero de configuración,
        // a parte de la configuración por defecto obtenida del fichero "hibernate.cfg.xml".
        // Primero indico el nombre del fichero de configuración,
        // y luego la clave por la que haré referencia a esa configuración.
        // Esto sólo habría que hacerlo una vez, en una fase de inicialización de la aplicación.
        HibernateMultipleConfig.addConfigFile("miOtraConfiguracion.hibernate.cfg.xml", "otraConfiguracion");

        // Trabajamos con la sesión por defecto de hibernate.
        // La sesión por defecto se obtiene del fichero "hibernate.cfg.xml".
        Session defaultSession = HibernateMultipleConfig.currentSession();

        // ...

        HibernateMultipleConfig.closeSession();

        // Trabajamos con una sesión concreta de hibernate.
        // La sesión se obtendrá a partir de la clave que se pasa como parámetro.
        Session aSession = HibernateMultipleConfig.currentSession("otraConfiguracion");

        // ...

        HibernateMultipleConfig.closeSession("otraConfiguracion");
    }
}

```

5. Conclusiones

Esta clase no pretende ser perfecta ni la solución definitiva (si encontráis algún bug decírmelo ;), pero si pretende ser un buen punto de partida para la gestión de múltiples bases de datos con Hibernate.

Seguramente un buen punto a mejorar es como se cierran las sesiones. Habría que intentar facilitar un poco más la vida al desarrollador, de forma que pudiera hacer algo como "aSession.close()" y ya se realizara toda la gestión (os lo dejo como deberes).

En cualquier caso no os olvidéis nunca de cerrar las sesiones (y en general de cerrar cualquier recurso: conexiones, streams, ...).

Espero que os sirva de ayuda, y nos vemos en el próximo tutorial.



[Puedes opinar sobre este tutorial aquí](#)

Recuerda

que el personal de [Autentia](#) te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#))

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos

Autentia = Soporte a Desarrollo & Formación

J2EE, EJBs, Struts...

[Autentia S.L.](#) Somos expertos en:

J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
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	<input type="text"/>
<input type="button" value="Enviar"/>	

Otros Tutoriales Recomendados ([También ver todos](#))

Nombre Corto	Descripción
Evitar doble-click en JSPs y Struts	Os mostramos como construir unas librerías de TAGs para evitar el problema de doble-click en aplicaciones JSP y como se soluciona (qué teneis que hacer) con el framework Struts
Otra implementación JDO con TJDO	Os mostramos como montar un ejemplo simple de JDO, a través de la implementación gratuita TJDO
Plantear una aplicación Web y Struts	Os mostramos un posible modo de plantear una aplicación Web (análisis) y darla forma. El FrameWork utilizado es struts y tratamos de identificar qué depende de este FrameWork y qué no.
CMP Entity Beans y MySql	Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql
Reingeniería JDO con Druid	Os mostramos como crear vuestras clases y descriptores JDO, de tablas existentes, con la herramienta gratuita Druid.
Novedades en Java 1.5	Ya está disponible la versión Beta del J2SDK 1.5. Os mostramos algunas de las nuevas características introducidas en el lenguaje Java: Clases genéricas, enumeraciones, bucles simplificados, etc.
Creación automática de recursos Hibernate con Middlegen	En este tutorial aprenderéis 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
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
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

[Framework desarrollo eclipse](#)

Aquí os mostramos algunas de las características de Eclipse

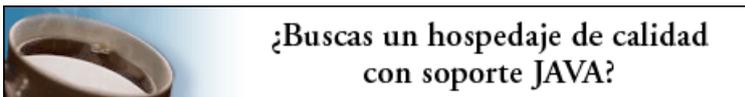
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

[Patrocinados por enredados.com Hosting en Castellano con soporte Java/J2EE](#)



www.AdictosAlTrabajo.com Optimizado 800X600