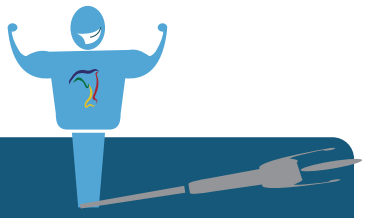
autentia

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...

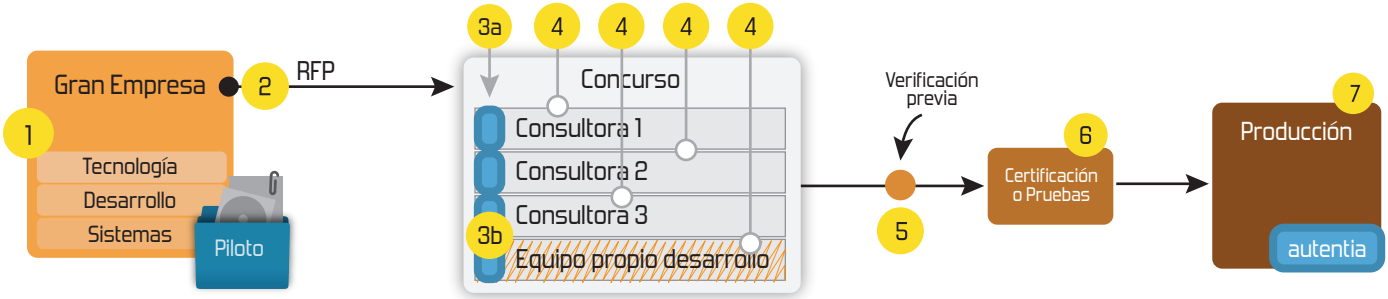
1. Desarrollo de componentes y proyectos a medida



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 conociemiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Empieza el jaleo: publicamos el libro
INFORMÁTICA PROFESIONAL
LAS REGLAS NO ESCRITAS PARA TRIUNFAR EN LA EMPRESA
-¿Crees que la informática es sólo progra
-¿Sabrías organizar eficientemente un e
-¿Quién te resuelve las dudas sobre esta profesión?..... o te siembra más?

Ultimas Noticias

» Creamos el grupo amigos de AdictosAlTrabajo en Facebook
» ¡¡¡¡ESPAAAAAAAAAAAAAAAAÑA!!!!
» VIII Charla Autentia - Alfresco Community Edition - Vídeos y Material
» IX Charla Autentia - Android
» Autentia Cycling Day
» La conferencia Agile-Spain2010 ya está aquí ¿Te la vas a perder?
» VIII Charla Autentia - Alfresco Community

+Noticias Destacadas

» Creamos el grupo amigos de AdictosAlTrabajo en Facebook
» VIII Charla Autentia - Alfresco Community Edition - Vídeos y Material
» IX Charla Autentia - Android
» La conferencia Agile-Spain2010 ya está aquí ¿Te la vas a perder?

+Comentarios Cómic

+Enlaces

Catálogo de servicios Autentia

Tríptico (6,3 MB)
Cómic (3,1 MB)

Tutorial desarrollado por

Enrique Viñé Lerma

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática por la Universidad Politécnica de Madrid.

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (3,1 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: mapeoVistasHibernate.pdf

Acceso de usuarios registrados:

E-mail:
Contraseña:
Entrar

Deseo registrarme

He olvidado mis datos de acceso

Registra tu empresa:

Descubre las ventajas de registrar tu empresa en AdictosAlTrabajo...

Registrar mi empresa

Listado de empresas ya registradas

Fecha de creación del tutorial: 2010-06-29

Share |

Mapeo de vistas con Hibernate

Índice de contenidos

1. Introducción
2. Clases del modelo
3. Configuración de Hibernate
4. Uso de @Subselect
5. Uso de <database-object>
    1. Alternativa utilizando @Subselect
    2. Alternativa utilizando un segundo <database-object>
6. Conclusiones

Google
Web
www.adictosaltrabajo.
Buscar

1. Introducción

Hibernate es una herramienta ORM que nos permite mapear nuestras clases del modelo con tablas de la B.D. ¿Pero qué es lo que ocurre si lo que queremos mapear es una vista en lugar de una tabla?. En principio no hay ningún problema, ya que podemos realizar el mapeo sobre la vista igual que si se tratase de una tabla.

El problema viene a la hora de utilizar la función "hbm2ddl" de Hibernate para generar nuestra B.D. automáticamente. Hibernate creará tablas para todas nuestras clases del modelo, independientemente de que nosotros pensemos en alguna de ellas como en una vista.

Evidentemente podríamos borrar las tablas innecesarias y crear las vistas correspondientes de forma manual, pero esto no vamos a poder hacerlo en algunos contextos. Por ejemplo, si tenemos una B.D. para los tests, la

Últimos tutoriales

2010-06-29
mapeoVistasHibernate

2010-06-29
Eclipse Helios, la nueva versión 3.6 de Eclipse

cual se generará cada vez que se lancen los mismos, Hibernate nos creará tablas en lugar de vistas. Y si estamos utilizando DBUnit para cargar los datos de prueba, deberemos añadir los datos de nuestras vistas (porque en realidad van a ser tablas normales).

Para intentar solucionar este problema, en este tutorial vamos a ver dos aproximaciones diferentes.

2. Clases del modelo

Para ambas aproximaciones vamos a utilizar las mismas clases del modelo: Profesor y Clase.

```
package com.autentia.tutorial;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.NotNull;

@Entity
public class Profesor implements Serializable {

        private static final long serialVersionUID = 4648652280201898240L;

        @Id
        @GeneratedValue(strategy = GenerationType.SEQUENCE)
        private Long id;

        @NotNull
        private String nombre;

        @NotNull
        private String apellidos;

        // Getters y Setters
        ...
}
```

Por cuestiones de legibilidad se han omitido los getters y los setters.

```
// Declaración de paquete e importaciones
...

@Entity
public class Clase implements Serializable {

        private static final long serialVersionUID = -5734276483633457611L;

        @Id
        @GeneratedValue(strategy = GenerationType.SEQUENCE)
        private Long id;

        @NotNull
        private Integer curso;

        @NotNull
        private Integer grupo;

        @NotNull
        private String descripcion;

        @NotNull
        private Integer numAlumnos;

        @ManyToOne
        @JoinColumn(name = "idTutor")
        @NotNull
        private Profesor tutor;

        // Getters y Setters
        ...
}
```

Cada clase tendrá, además de otros campos, un número de alumnos y un tutor (de tipo Profesor). Imaginemos que en nuestra aplicación vamos a querer obtener frecuentementeel número de alumnos que tiene a cargo cada tutor. Podríamos tener una vista que obtuviese los datos correspondientes:

```
create or replace view AlumnosTutor as
select c.idTutor, sum(c.numAlumnos) totalAlumnos
from Clase c
group by c.idTutor
order by c.idTutor
```

Para poder obtener desde nuestro código los valores de esta vista, tendremos otra entidad en nuestro modelo:

```
// Declaración de paquete e importaciones
...

@Entity
public class AlumnosTutor implements Serializable {
```


2010-06-28
Mac Automator y como renombrar múltiples ficheros

2010-06-21
Prototipado rápido de pantallas (mocks) con Google Drawings

2010-06-17
IAQ (Interesting Asked Questions), SPI ¿qué es, hay que usarlo, o no, cuándo?

2010-06-14
Instalar PostgreSQL en Mac OS X Snow Leopard

2010-06-14
Integrando tus redes sociales con HootSuite

2009-02-26
Edición de vídeo para publicar en YouTube (II)

2010-06-14
Primeros pasos en comunidad

2010-06-11
Cómo analizar con Sonar un proyecto sin maven

2010-06-11
DBUnit y aplicaciones JDBC

2010-06-11
Edición de vídeo para publicar en YouTube (I)

2010-06-10
Generación de Informes con JasperReports en PHP

2010-06-07
Introducción a BPMN

2010-06-07
Instalación Intalio Server

2010-05-31
JSF 2 Ajax y Bean Validation

2010-05-27
Instalación Intalio Designer

2010-05-26
Generar facturas electrónicas de manera gratuita

```
        private static final long serialVersionUID = 1226577089185303114L;

        @Id
        @Column(insertable = false, updatable = false)
        private Long idTutor;

        @OneToOne
        @JoinColumn(name = "idTutor", insertable = false, updatable = false)
        private Profesor tutor;

        @Column(insertable = false, updatable = false)
        private Integer totalAlumnos;


        // Getters y Setters
        ...
}
```

Esta clase contendrá un tutor y su número de alumnos. Se han marcado todas las columnas con "insertable" y "updatable" a false, ya que en teoría es una vista y no tendremos que insertar ni actualizar nada en ella.

3. Configuración de Hibernate

Si estamos utilizando Spring en nuestro proyecto, deberemos añadir la configuración de Hibernate en el "applicationContext.xml". Si queréis saber más sobre cómo configurar una aplicación que utilice Spring, Hibernate y anotaciones, os recomiendo consultar este tutorial de Alex.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:util="http://www.springframework.org/schema/util" xmlns:context="http://www.springframework.org/sche
        xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="
                http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-bea
                http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-
                http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-2.5.xsd
                http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5
                http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xs

        <context:component-scan base-package="com.autentia" />

        <tx:annotation-driven transaction-manager="transactionManager" />

        <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
                <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
                <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
                <property name="username" value="username" />
                <property name="password" value="********" />
        </bean>

        <bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
                <property name="dataSource" ref="dataSource" />
                <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />
                <property name="configLocation">
                        <value>classpath:hibernate.cfg.xml</value>
                </property>
                <property name="hibernateProperties">
                        <props>
                                <prop key="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</prop>
                                <prop key="hibernate.hbm2ddl.auto">create-drop</prop>
                                <prop key="hibernate.show_sql">update</prop>
                        </props>
                </property>
        </bean>

        <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
                <property name="sessionFactory" ref="sessionFactory" />
        </bean>
</beans>
```

En mi caso utilizaré una B.D. Oracle Database 10g Express Edition, en mi propia máquina. Hay que fijarse en la propiedad "hibernate.hbm2ddl.auto" que se ha puesto a "create-drop", para que cada vez que arranquemos la aplicación se creen las tablas de la B.D. desde cero y después se borren. Esto sólo es recomendable para hacer pruebas, después de lo cual sería conveniente cambiar el valor por "update".

Además, estamos haciendo referencia al fichero "hibernate.cfg.xml", en el que indicamos las clases mapeadas por Hibernate:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
        <session-factory>
                <property name="current_session_context_class">thread</property>

                <mapping class="com.fia.auditorium.model.Clase" />
                <mapping class="com.fia.auditorium.model.Profesor" />
```


2010-05-25
Atajos de teclado más utilizados para Eclipse

2010-05-24
Botones para añadir eventos directamente a calendario (Google Calendar y Outlook)

2010-05-24
Procesar un fichero EDI con Talend

2010-05-17
Añadir botones para social bookmarking y sharing service.

2010-05-17
Intercomunicación entre portlets.

2010-05-17
Actualización de bugzilla a la versión 3.6.

2010-05-11
eCobertura y EclEMMA: Plugins para la medición de cobertura del código.

2010-05-11
VirtualBox. Configuración de la conexión de red.

2010-05-11
Múltiples datasources en JasperReports

2010-05-10
Utilización de arquetipos y plugins de LifeRay para Maven.

2010-05-07
Instalar un componente en Talend Open Studio (TOS)

2010-05-06
Talend Open Studio (TOS) 4.0

2010-05-05
Gestión de los Requisitos (y II) : los casos de uso

2010-05-05
Gestión de los requisitos

2010-05-04
Declaración de IVA trimestral en la AEAT por Internet

2010-05-04

```
                <mapping class="com.fia.auditorium.model.AlumnosTutor" />
        </session-factory>
</hibernate-configuration>
```

4. Uso de @Subselect

Lo que hemos visto hasta ahora no es nada nuevo y, si dejamos así las cosas, nuestra aplicación funcionará sin problemas. Pero si Hibernate está generando nuestras tablas automáticamente, nos generará "AlumnosTutor" como una tabla y no como una vista.

Nuestra primera alternativa para evitar esto, es utilizar la anotación @Subselect para mapear la entidad AlumnosTutor a una subselect de SQL. De esta manera, la entidad será de sólo lectura y no buscará los datos en ninguna tabla, sino que los sacará de la propia consulta proporcionada.

Sólamente tendremos que añadir la anotación @Subselect en nuestra clase AlumnosTutor, pasándole como valor la consulta SQL:

```
...

@Entity
@Subselect("select c.idTutor, sum(c.numAlumnos) totalAlumnos from Clase c group by c.idTutor order by c.idTutor")
public class AlumnosTutor implements Serializable {

...
```

De esta manera, en nuestro esquema no se creará la tabla AlumnosTutor, aunque tampoco se creará la vista. Cada vez que queramos recuperar datos de esta entidad, se ejecutará una consulta que obtendrá los datos de la subconsulta que hemos indicado.

Este método tiene la ventaja de que podemos usarlo aunque tengamos un esquema de B.D. que no podamos modificar para crear la vista que necesitamos.

5. Uso de <database-object>

Como se indica en la documentación de Hibernate, los objetos auxiliares de base de datos permiten hacer CREATE y DROP de objetos arbitrarios de B.D. De esta manera, podemos definir completamente un esquema a partir de los ficheros de mapeo de Hibernate.

A pesar de estar específicamente diseñados para crear o borrar cosas como triggers o procedimientos, es válido cualquier comando SQL que pueda ser ejecutado por medio del método "java.sql.Statement.execute()" (por ejemplo, ALTERs, INSERTs, etc.).

Lo que vamos a hacer nosotros, es aprovechar esta configuración para crear nuestra vista en B.D. Para ello crearemos el fichero "hibernate.mapping.xml" que colocaremos en la misma carpeta que el fichero de configuración de Hibernate.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
        <database-object>
                <create>
                        create or replace view AlumnosTutor as
                        select c.idTutor, sum(c.numAlumnos) totalAlumnos
                        from Clase c
                        group by c.idTutor
                        order by c.idTutor
                </create>
                <drop>drop view AlumnosTutor</drop>
                <dialect-scope name='org.hibernate.dialect.Oracle10gDialect' />
        </database-object>
</hibernate-mapping>
```

Cuando Hibernate genere nuestra B.D. ejecutará la sentencia indicada en <create>, aunque sólamente en el caso de que el dialecto utilizado corresponda con el indicado en <dialect-scope>.

Debemos referenciar el fichero anterior desde "hibernate.cfg.xml", añadiendo la siguiente línea dentro del elemento <session-factory>:

```
<mapping resource="hibernate.mapping.xml" />
```

Además, tendremos que quitar la anotación @Subselect de nuestra clase AlumnosTutor.

Si ahora dejamos que Hibernate genere nuestras tablas... ¡tendremos un problema! Los <database-object> se ejecutarán siempre después de crear las tablas correspondientes a las entidades que tenemos mapeadas, por lo que Hibernate generará primero la tabla AlumnosTutor y fallará al intentar crear una vista con el mismo nombre.

Después de intentar varias soluciones, como el uso de @Immutable y algunas otras anotaciones de Hibernate sin resultados positivos, me he quedado con dos alternativas que sí funcionan.

5.1 Alternativa utilizando @Subselect

Antes de nada debo advertir que esta alternativa no es recomendable en absoluto: simplemente funciona y por eso la menciono. Se trata de dejar todo lo anterior como está, pero incluir una anotación @Subselect de nuevo en nuestra clase AlumnosTutor:

```
...
```


Certificados en Firefox (FNMT y AEAT)

2010-04-26
JCaptcha - Generación de Captchas en Java

2010-04-23
Instalar Puente PHP-Java en Tomcat

2010-04-22
AppWidget Android: Ejemplo usando BroadcastReceiver y Localización

2010-04-20
Facelets en JSF 2: sistema de plantillas y componentes por composición.

2010-04-19
DbVisualizer free version.

2010-04-09
Session TimeOut en RichFaces, con el soporte de Jboss Seam.

2010-04-08
Jetspeed-2 de Apache Software Foundation

2010-04-07
Primeros pasos con Balsamiq Mockups

2010-03-18
Revisando los ejemplos de Cocos2d para IPhone.

2010-03-16
Organización de eventos con StageHQ

2010-03-15
Retrasar la carga de Javascript con jQuery.getScript().

2010-03-15
Optimización de páginas web con Page Speed.

2010-03-09
JSF 2 ya está aquí !!! The JSF Return, ahora más sencillo que nunca !!!

2010-03-08
Instalación de tus programas en tu IPhone.

2010-03-04

```
@Entity
@Subselect("select * from AlumnosTutor")
public class AlumnosTutor implements Serializable {

...
```

De esta manera le decimos a Hibernate que no cree una tabla para esta entidad, sino que utilice los valores obtenidos de la consulta (que lo que hace es obtener los valores de la vista). Lo malo de esta opción es que cada vez que queramos recuperar datos de esta entidad, se ejecutará una consulta con un subselect. En realidad estamos "haciendo trampa", engañando a Hibernate para que utilice una subconsulta cuando en realidad sí que vamos a tener una vista en la B.D.

5.2 Alternativa utilizando un segundo <database-object>

Esta alternativa tampoco es perfecta, pero es más recomendable que la anterior y que otras que he probado. Lo que vamos a hacer, ya que no podemos evitar que Hibernate nos cree la tabla antes que nuestra vista, es forzarle a borrar de nuevo la tabla.

Esta vez tenemos que modificar únicamente "hibernate.mapping.xml", para que quede de la siguiente manera:

```
<hibernate-mapping>
        <database-object>
                <create>
                        drop table AlumnosTutor
                </create>
                <drop></drop>
                <dialect-scope name='org.hibernate.dialect.Oracle10gDialect' />
        </database-object>

        <database-object>
                <create>
                        create or replace view AlumnosTutor as
                        select c.idTutor, sum(c.numAlumnos) totalAlumnos
                        from Clase c
                        group by c.idTutor
                        order by c.idTutor
                </create>
                <drop>drop view AlumnosTutor</drop>
                <dialect-scope name='org.hibernate.dialect.Oracle10gDialect' />
        </database-object>
</hibernate-mapping>
```

De esta manera, Hibernate generará tablas para todas nuestras entidades y, después, creará los "database-object" indicados por orden (ejecutará las siguientes sentencias SQL):

- drop table AlumnosTutor
- create or replace view AlumnosTutor as ...

Al final en nuestro esquema tendremos únicamente las tablas Profesor y Clase, y la vista AlumnosTutor. Reconozco que no es la manera más elegante del mundo, ya que sería preferible evitar que Hibernate crease inicialmente la tabla AlumnosTutor. Pero eso es algo que no he conseguido, por más pruebas que he podido hacer.

6. Conclusiones

- La generación automática de nuestro esquema de B.D. es una característica muy potente y útil de Hibernate. Si queremos que nuestras vistas se generen también de forma automática, deberemos añadir un poco de configuración adicional.
- Si necesitamos crear vistas pero no podemos tocar el esquema de la B.D., el uso de @Subselect nos permitirá mapear una entidad utilizando una subconsulta.
- Si tenemos la posibilidad de crear la vista en B.D. es preferible hacerlo utilizando los elementos <database-object> que permiten ejecutar sentencias sobre la propia B.D.
- Al utilizar <database-object> casi siempre dependeremos del dialecto de la B.D. utilizado, por lo que si cambiamos de dialecto, deberemos añadir nuevos <database-object> para poder soportarlos.

Share |

¿Qué te ha parecido el tutorial? Déjanos saber tu opinión y ¡vota!

Muy malo Malo Regular Bueno Muy bueno

Votar

(Sólo para usuarios registrados)

» Registrate y accede a esta y otras ventajas «

Sacar Release de un proyecto con Maven

2010-03-03
Instalación de Subversion y Apache en Ubuntu

Últimas ofertas de empleo

2010-04-28
Comercial - Compras - CORDOBA.

2010-04-25
Otras Sin catalogar - MADRID.

2010-04-25
Atención a cliente - Call Center - MADRID.

2010-04-21
Comercial - Ventas - MADRID.

2009-06-25
Atención a cliente - Call Center - BARCELONA.

Ads by Google

Cursos Gratis
Hibernate Criteria
Guitarra
Hibernate Examples

Anímate y coméntanos lo que pienses sobre este tutorial

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» Registrate y accede a esta y otras ventajas «

Autor	Mensaje de usuario registrado

- Puedes inscribirte en nuestro servicio de notificaciones 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

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

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.

¿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?

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

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf

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.