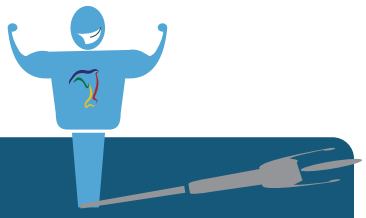
autentia

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...

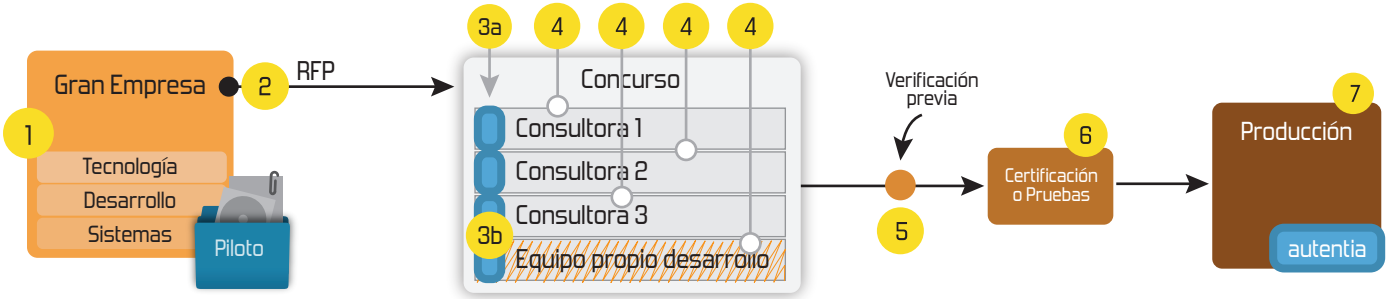
1. Desarrollo de componentes y proyectos a medida



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

Powered by autentia Adictos AL Trabajo ¿Hay felicidad sin tristeza? Ante la crisis... esfuerzo y esperanza

Powered by autentia Hosting patrocinado por enredados

Inicio Quienes somos Tutoriales Formación Colabora Comunidad Comic Charlas Más

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

Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución

[¡NUEVO!] 2008-12-01

2008-11-17

2008-09-01

2008-07-31

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

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 (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: eventosenhibernatei.pdf

Fecha de creación del tutorial: 2009-01-27

Eventos en Hibernate (Parte I)

Índice de contenidos

1. Introducción

Este es el primero de una serie de tutoriales en los que vamos a hablar del manejo de eventos en Hibernate. En este primer tutorial nos vamos a centrar en los oyentes (listeners) que pueden utilizarse con una SessionFactory para capturar los eventos que se producen al cargar una entidad, guardarla, actualizarla, borrarla, etc. y realizar algún tipo de acción en ese momento. En el próximo tutorial veremos cómo asociar métodos de retrollamada en nuestras entidades a determinados eventos, de manera que éstos métodos se ejecutarán al producirse los eventos correspondientes. Esto útlimo además puede hacerse por medio de anotaciones que siguen la especificación de EJB3.

2. Entorno

Para la realización de este tutorial se han utilizado las siguientes herramientas:

- Hardware: Portátil Asus G50Vseries (Core Duo P8600 2.4GHz, 4GB RAM, 320 GB HD).
- Sistema operativo: Windows Vista Ultimate.
- Eclipse Ganymede 3.4.1, con el plugin Q (http://code.google.com/p/q4e) para Maven
- JDK 1.6.0
- Maven 2.0.9
- MySql 5.1.30
- Hibernate 3

3. Preparando una aplicación de ejemplo

3.1. Usando Maven para crear el proyecto

Catálogo de servicios Autentia (PDF 6,2MB)

En formato comic...

Google

Web

www.adictosaltrabajo.com

Buscar

Últimos tutoriales

2009-01-27
Eventos en Hibernate (parte I)

2009-01-25
Aprendiendo XMLSchema a través de ejemplos

2009-01-20
Pruebas Software con Junit 4 y Eclipse

2009-01-19
Executor : Un programa para ejecutarlos a todos.

2009-01-18
Soap Monitor: Monitorización de mensajes SOAP en Axis2

2009-01-16
Restaurar una Base de Datos en SQL Server o como cambiar el propietario de los objetos de la base de datos

2009-01-14
Solución a NoClassDefFoundError: SWTResourceUtil

2009-01-14
Desarrollo de aplicaciones Web con Struts 1

2009-01-07
Log4J: Cómo crear un log que trabaje hacia una Base de Datos.

2009-01-05
Introducción a Google Chart API

Últimas ofertas de empleo

2008-12-22
Otras - Mecánica - SEVILLA.

2008-11-27
Comercial - Ventas - ALICANTE.

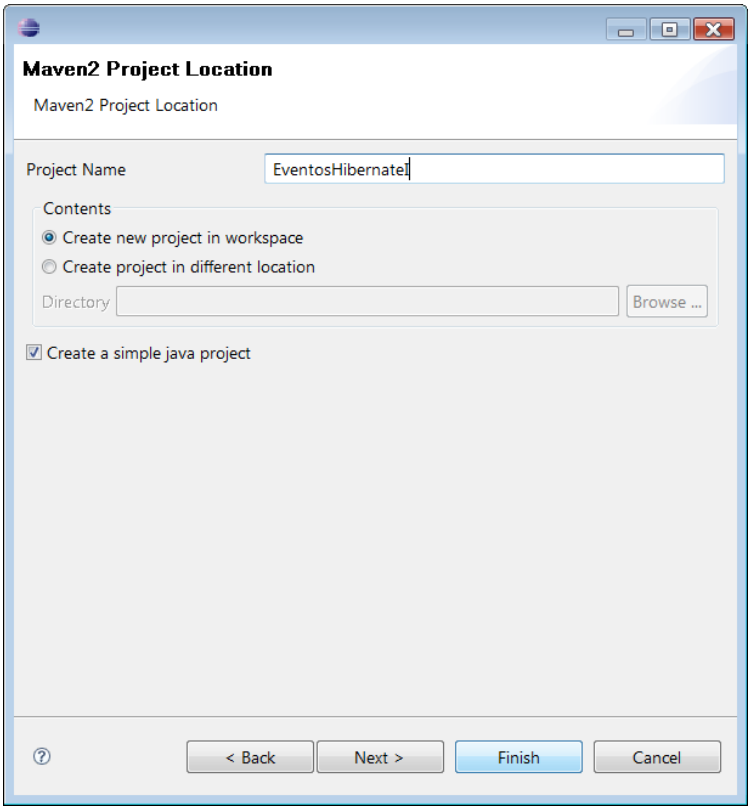
2008-10-30
Comercial - Ventas - BARCELONA.

2008-10-30
T. Información - Analista / Programador - BARCELONA.

Anuncios Google Informatica Facil Hibernate Java JDBC Examples JSP J2EE Cursos Informatica

Para mostrar el uso de los oyentes de eventos vamos a realizar una sencilla aplicación Java que introduzca una serie de usuarios en una base de datos y que después los recupere y los muestre por la salida estándar.

Creamos un nuevo proyecto Maven, al que llamaremos EventosHibernateI, marcamos la casilla para crear un simple proyecto Java y pulsamos finalizar.



Editamos el fichero pom.xml para añadir las dependencias necesarias para utilizar Hibernate y MySql. Además, cambiamos el nivel de compilación por defecto utilizado por Maven para poder utilizar anotaciones con Hibernate.

2008-10-27
T. Información - Analista / Programador - CIUDAD REAL.

Anuncios Google

view plain print ?

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd" >
    <modelVersion>4.0.0</modelVersion>
    <groupId>EventosHibernateI</groupId>
    <artifactId>EventosHibernateI</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>EventosHibernateI</name>
    <url>http://maven.apache.org</url>

    <!--
        Cambia el nivel de compilacion por defecto para poder utilizar anotaciones
    -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.5</source>
                    <target>1.5</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <!-- JUnit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.4</version>
            <scope>test</scope>
        </dependency>

        <!-- MySql Connector -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.0.5</version>
        </dependency>

        <!-- log4j -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.5.2</version>
            <scope>test</scope>
        </dependency>

        <!-- Hibernate -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>3.3.1.GA</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-annotations</artifactId>
            <version>3.4.0.GA</version>
        </dependency>
        <dependency>
            <groupId>javassist</groupId>
            <artifactId>javassist</artifactId>
            <version>3.8.1.GA</version>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
</project>
```

3.2 Creando el modelo

Ahora crearemos nuestro modelo, consistente de una única entidad Usuario, la cual será manejada por Hibernate para persistir los usuarios en la base de datos. La entidad tendrá los campos nombre, apellidos, fecha de nacimiento y edad.

El campo edad se marca como Transient para que Hibernate no lo guarde en la base de datos. Este campo será calculado mediante la función "calcularEdad" en base al valor de la fecha de nacimiento. Más adelante veremos como conseguir que está función sea llamada automáticamente al producirse el evento PostLoad, que ocurre justo después de la carga de los datos de la entidad.

view plain print ?

```
01. package modelo;
02.
03. import java.util.Calendar;
04. import java.util.Date;
05. import java.util.GregorianCalendar;
06. import javax.persistence.Entity;
07. import javax.persistence.GeneratedValue;
08. import javax.persistence.Id;
09. import javax.persistence.Transient;
10.
11. @Entity
12. public class Usuario {
13.
14.     @Id
15.     @GeneratedValue
16.     private Integer id;
17.
18.     private String nombre;
19.
20.     private String apellidos;
21.
22.     private Date nacimiento;
23.
24.     @Transient
25.     private int edad;
26.
27.     public Usuario() {
28.         super();
29.     }
30.
31.     public Usuario(String nombre, String apellidos) {
32.         this.nombre = nombre;
33.         this.apellidos = apellidos;
34.     }
35.
36.     public void calcularEdad() {
37.         Calendar cumple = new GregorianCalendar();
38.         Calendar ahora = new GregorianCalendar();
39.         cumple.setTime(nacimiento);
40.         ahora.setTime(new Date()); // Comprobar que la fecha de nacimiento sea
41.                                    // anterior a la actual
42.         if (ahora.compareTo(cumple) < 0) { // Si no es anterior poner la edad a
43.                                            // 0
44.             edad = 0;
45.         } else {
46.             int ajuste = (ahora.get(Calendar.DAY_OF_YEAR)
47.                     - cumple.get(Calendar.DAY_OF_YEAR) < 0) ? -1 : 0;
48.             edad = ahora.get(Calendar.YEAR) - cumple.get(Calendar.YEAR)
49.                     + ajuste;
50.         }
51.     }
52.
53.     public String getNombre() {
54.         return nombre;
55.     }
56.
57.     public void setNombre(String nombre) {
58.         this.nombre = nombre;
59.     }
60.
61.     public String getApellidos() {
62.         return apellidos;
63.     }
64.
65.     public void setApellidos(String apellidos) {
66.         this.apellidos = apellidos;
67.     }
68.
69.     public Date getNacimiento() {
70.         return nacimiento;
71.     }
72.
73.     public void setNacimiento(Date nacimiento) {
74.         this.nacimiento = nacimiento;
75.         calcularEdad();
76.     }
77.
78.     public int getEdad() {
79.         return edad;
80.     }
81.
82.     public void setEdad(int edad) {
83.         this.edad = edad;
84.     }
85.
86.     public Integer getId() {
87.         return id;
88.     }
89. }
```

Nota: deberemos cambiar el nivel de compilación en Eclipse como mínimo al 1.5, si no lo está ya, para soportar las anotaciones.

Ahora que tenemos una bonita entidad para persistir, vamos a crearnos un simple DAO que nos permita guardar un usuario en la base de datos y recuperar la lista de usuarios.

view plain print ?

```
01. package modelo;
02.
03. import java.util.ArrayList;
04. import java.util.List;
05.
06. import org.hibernate.Session;
07. import org.hibernate.SessionFactory;
08. import org.hibernate.Transaction;
09. import org.hibernate.cfg.AnnotationConfiguration;
10.
11. public class UsuarioDao {
12.
13.     private static SessionFactory sf = new AnnotationConfiguration().configure( "hibernate.cfg.xml").buildSessionFactory();
14.
15.     public void guardarUsuario(Usuario usuario) {
16.         Session sesion = sf.openSession();
17.         Transaction tx = null;;
18.
19.         try {
20.             tx = sesion.beginTransaction();
21.             sesion.save(usuario);
22.             tx.commit();
23.         } catch (Exception e) {
24.             tx.rollback();
25.         } finally {
26.             sesion.close();
27.         }
28.     }
29.
30.     public List<Usuario> buscarUsuarios() {
31.         List<Usuario> usuarios = new ArrayList<Usuario>();
32.
33.         Session sesion = sf.openSession();
34.         Transaction tx = null;;
35.
36.         try {
37.             tx = sesion.beginTransaction();
38.             usuarios = sesion.createQuery( "from Usuario").list();
39.             tx.commit();
40.         } catch (Exception e) {
41.             tx.rollback();
42.         } finally {
43.             sesion.close();
44.         }
45.
46.         return usuarios;
47.     }
48. }
```

Añadimos dentro de la carpeta resources el fichero de configuración de Hibernate.

view plain print ?

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <!DOCTYPE hibernate-configuration PUBLIC
03.         "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
04.         "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd" >
05. <hibernate-configuration>
06.     <session-factory>
07.         <property name="hibernate.dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
08.         <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
09.         <property name="hibernate.connection.username">root</property>
10.         <property name="hibernate.connection.password">root</property>
11.         <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/eventos?autoReconnect=true</property>
12.
13.         <property name="hibernate.hbm2ddl.auto">update</property>
14.         <property name="hibernate.show_sql">true</property>
15.
16.         <mapping class="modelo.Usuario"/>
17.     </session-factory>
18. </hibernate-configuration>
```

Hemos añadido la propiedad **hibernate.hbm2ddl.auto** con el valor auto para que Hibernate cree de manera automática las tablas en caso de que no existan. Eso sí, no debemos olvidar crear primero el esquema eventos en la base de datos.

3.3 Creando clases de prueba y probando la aplicación

Ahora crearemos un par de clases para probar el funcionamiento de nuestra aplicación. La primera clase guardará una serie de usuarios en la base de datos. La segunda recuperará la lista de usuarios y mostrará por la consola su nombre, apellidos y edad.

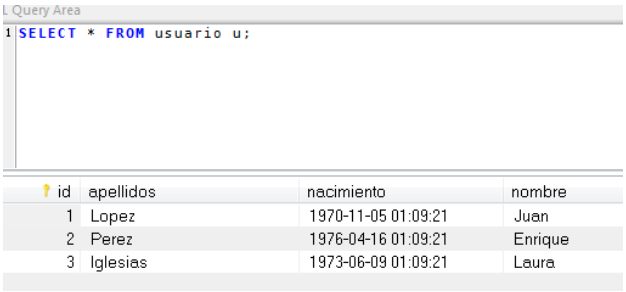
Aquí está el código de nuestra clase PruebaInsercion.

```
view plain print ?
01. package prueba;
02.
03. import java.util.Calendar;
04. import java.util.Date;
05. import java.util.GregorianCalendar;
06.
07. import modelo.Usuario;
08. import modelo.UsuarioDao;
09.
10. public class PruebaInsercion {
11.     public static void main(String[] args) {
12.         Calendar cal = new GregorianCalendar();
13.         cal.set(1970, 10, 5);
14.         nuevoUsuario("Juan", "Lopez", cal.getTime());
15.         cal.set(1976, 3, 16);
16.         nuevoUsuario("Enrique", "Perez", cal.getTime());
17.         cal.set(1973, 5, 9);
18.         nuevoUsuario("Laura", "Iglesias", cal.getTime());
19.     }
20.
21.     private static void nuevoUsuario(String nombre, String apellidos, Date edad) {
22.         UsuarioDao dao = new UsuarioDao();
23.         Usuario usuario = new Usuario(nombre, apellidos);
24.         usuario.setNacimiento(edad);
25.         dao.guardarUsuario(usuario);
26.     }
27. }
```

Y aquí tenemos la clase PruebaListado, que mostrará el nombre, apellidos y edad de los usuarios obtenidos de la base de datos.

```
view plain print ?
01. package prueba;
02.
03. import java.util.List;
04.
05. import modelo.Usuario;
06. import modelo.UsuarioDao;
07.
08. public class PruebaListado {
09.     public static void main(String[] args) {
10.         UsuarioDao dao = new UsuarioDao();
11.         List<Usuario> usuarios = dao.buscarUsuarios();
12.         for (Usuario usuario : usuarios) {
13.             System.out.printf("%s %s tiene %s años\n", usuario.getNombre(),
14.                     usuario.getApellidos(), usuario.getEdad());
15.         }
16.     }
17. }
```

Ejecutamos primero la inserción y comprobamos que se han añadido correctamente los usuarios a nuestra base de datos:



id	apellidos	nacimiento	nombre
1	Lopez	1970-11-05 01:09:21	Juan
2	Perez	1976-04-16 01:09:21	Enrique
3	Iglesias	1973-06-09 01:09:21	Laura

Ahora ejecutamos la prueba de listado y observamos... ¡Cáspita! O todos nuestros usuarios son recién nacidos o creo que hemos cometido algún error ;)

```
Juan Lopez tiene 0 años
Enrique Perez tiene 0 años
Laura Iglesias tiene 0 años
```

Efectivamente, dijimos que íbamos a hacer que la función "calcularEdad" se disparase de forma "automática" al cargar nuestra entidad. ¿Y no era este un tutorial de eventos en Hibernate?.

Hasta ahora sólamente hemos visto el uso normal de Hibernate para persistir entidades. A continuación vamos a ver cómo configurar un oyente para que se ejecute al dispararse un determinado evento.

4. Creando un oyente para escuchar

Una vez preparada nuestra aplicación sólamente nos resta crear una clase oyente que se ejecute al cargar cada entidad Usuario. El oyente se encargará de actualizar el valor de la edad del usuario en función de su fecha de nacimiento.

Los oyentes en Hibernate son en la práctica singletons, por lo que pueden ser compartidos entre peticiones y por tanto no deberían guardar ningún tipo de estado.

view plain print ?

```
01. package oyentes;
02.
03. import modelo.Usuario;
04.
05. import org.hibernate.event.PostLoadEvent;
06. import org.hibernate.event.PostLoadEventListener;
07.
08. public class PostLoadOyente implements PostLoadEventListener {
09.
10.     private static final long serialVersionUID = 3672646295810997521L;
11.
12.     public void onPostLoad(PostLoadEvent event) {
13.         System.out.println("Ejecutando nuestro oyente");
14.         if (event.getEntity() instanceof Usuario) {
15.             Usuario usuario = (Usuario) event.getEntity();
16.             usuario.calcularEdad();
17.         }
18.     }
19. }
```

Nuestro implementa la interfaz PostLoadEventListener, para capturar este tipo de evento. También podríamos haber extendido la clase PostLoadEvent, el oyente predeterminado de Hibernate.

En Hibernate existem básicamente eventos por cada método de la clase Session (guardar, borrar, buscar...). A continuación se muestra una tabla con todos los eventos existentes:

Evento	
Delete	PostInsert
DirtyCheck	PostLoad
Evict	PostUpdate
Flush	PreDelete
InitializeCollection	PreInsert
Load	PreLoad
Lock	PreUpdate
Merge	Refresh
Persist	Replicate
PostDelete	SaveOrUpdate

Por cada tipo de evento existe una interfaz, de la forma ***evento*EventListener** y una clase base de Hibernate de la forma ***evento*Event**. Tanto las interfaces como las clases se encuentran dentro del paquete "org.hibernate.event".

El último paso consiste en asignar el oyente que hemos programado al evento PostLoad, para que se ejecute justo después de cargar las entidades. En nuestro caso sólo existe una entidad, pero un oyente escuchará un evento determinado independientemente de la entidad sobre la que ocurra el mismo. Por tanto debemos ser cuidadosos al utilizar nuestro oyente, asegurándonos de que el evento haya sido producido por la entidad que creemos.

Para configurar nuestro oyente deberemos añadir las siguientes líneas al fichero de configuración de hibernate, dentro de la etiqueta <session-factory>.

view plain print ?

```
01. <event type="post-load">
02.   <listener class="oyentes.PostLoadOyente"/>
03.     <listener class="org.hibernate.event.def.DefaultPostLoadEventListener" />
04. </event>
```

Si volvemos a ejecutar la clase que muestra el listado, esta vez obtendremos el resultado esperado:

```
Hibernate: select usuario0_.id a
Ejecutando nuestro oyente
Ejecutando nuestro oyente
Ejecutando nuestro oyente
Juan Lopez tiene 38 años
Enrique Perez tiene 32 años
Laura Iglesias tiene 35 años
```

Esta vez el oyente se ha ejecutado justo después de que se carguen los datos de cada usuario, actualizando el campo edad de los mismos.

5. Dos mejor que uno

¿Y qué pasa si queremos tener varios oyentes para un mismo evento? Simplemente tendremos que añadir el oyente a la pila de oyentes en el fichero de configuración de hibernate. Todos los oyentes serán llamados en su orden de aparición; por esta razón es aconsejable mantener siempre al final de la lista el oyente predeterminado de Hibernate, que podría realizar acciones adicionales.

El fichero de configuración quedaría ahora de la siguiente manera:

view plain print ?

```
01. <event type="post-load">
02.   <listener class="oyentes.PostLoadOyente"/>
03.   <listener class="oyentes.LiftingOyente"/>
04.   <listener class="org.hibernate.event.def.DefaultPostLoadEventListener" />
05. </event>
```

Y, por supuesto, deberemos añadir nuestra clase oyente:

view plain print ?

```
01. package oyentes;
02.
03. import modelo.Usuario;
04.
05. import org.hibernate.event.PostLoadEvent;
06. import org.hibernate.event.PostLoadEventListener;
07.
08. public class LiftingOyente implements PostLoadEventListener {
09.
10.     private static final long serialVersionUID = -6982862192025601114L;
11.
12.     public void onPostLoad(PostLoadEvent event) {
13.         if (event.getEntity() instanceof Usuario) {
14.             Usuario usuario = (Usuario) event.getEntity();
15.             System.out.println( "Haciendo lifting a " + usuario.getNombre());
16.             usuario.setEdad(usuario.getEdad() -  3);
17.         }
18.     }
19. }
```

Este nuevo oyente aplicará un "lifting" a los usuarios, rejuveneciéndoles 3 años (¡qué maravilla!). Si ejecutamos de nuevo el listado observaremos cómo ha descendido la media de edad de los usuarios.

```
Ejecutando nuestro oyente
Haciendo lifting a Juan
Ejecutando nuestro oyente
Haciendo lifting a Enrique
Ejecutando nuestro oyente
Haciendo lifting a Laura
Juan Lopez tiene 35 años
Enrique Perez tiene 29 años
Laura Iglesias tiene 32 años
```

6. Conclusiones

A la vista de los resultados de este tutorial podemos extraer algunas conclusiones:

- El uso de oyentes nos permite realizar acciones asociadas a la persistencia de nuestros objetos de manera automática y transparente a nuestra lógica de negocio. Esto nos permite realizar acciones que de otra manera tendríamos que controlar manualmente cada vez que hiciésemos una llamada a la Session de Hibernate.
- Podemos asignar varios oyentes por cada evento para que realizen diferentes tareas y añadir o quitar oyentes de manera sencilla por medio del fichero de configuración de Hibernate.
- Deberemos implementar una clase por cada oyente que necesitemos. Estas clases deben implementar la interfaz correspondiente y distinguir en su código de alguna manera la entidad sobre la que se está produciendo el evento.

Ya hemos visto el poder que tienen los eventos para realizar acciones asociadas con la persistencia y alterar los valores de los campos de nuestras entidades. No obstante puede resultar un poco complicado tener que implementar una clase por cada oyente si simplemente lo que queremos es ejecutar un método de la entidad que realice ciertas operaciones sobre sus campos. ¿No sería más sencillo poner simplemente una anotación en el método que queramos ejecutar al producirse un evento? Responderemos a esta y otras preguntas en la segunda parte de este tutorial.

Y eso es todo. En Autentia siempre estamos buscando soluciones para los problemas que encontramos en nuestro trabajo cotidiano, y esperamos que este y otros de nuestros tutoriales os ayuden también a vosotros a solucionar problemas similares.

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

Muy malo Malo Regular Bueno Muy bueno

Votar

- 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

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

soluciones reales para su negocio

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	Valoración	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	8780	-	pdf
Hibernate y Joins con la clase Criteria	En este tutorial vamos a ver como hacer "joins" entre entidades relacionadas, y las implicaciones que esto puede tener, usando Hibernate	2007-06-25	6885	-	pdf
Como implementar el método equals(Object) en objetos persistentes de Hibernate, y otras consideraciones.	En este tutorial Alejandro Pérez nos muestra cómo implementar el método equals(Object) en objetos persistentes de Hibernate, y otras consideraciones.	2008-02-05	2406	-	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	63318	-	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	6232	-	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	14761	-	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	10596	-	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	20768	-	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	18015	-	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	8576	-	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.