

¿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)

Estas en: [Inicio](#) [Tutoriales](#) AOP con AspectJ y Maven

Últimas Noticias

- » Historia de la informática. Capítulo 65. 1987
- » Historia de la informática. Capítulo 64. 1986
- » Autentia en la Sun Open Communities Forum
- » Comentario del libro: El economista naturalista de Robert Frank
- » Contratos ágiles: Vendiendo Scrum a tus clientes.
- » Resumen de la cuarta charla gratuita de Autentia: SCRUM (con video)
- » Si se pregunta ¿Qué ofrece este Web?
- » Vota AdictosAlTrabajo.com en DZone

+Noticias Destacadas

- » Contratos ágiles: Vendiendo Scrum a tus clientes.
- » Quinta charla Autentia + Proyectalis + Agile Spain: Contratos ágiles: Vendiendo Scrum a tus clientes
- » Lo mejor de esta semana: Curso de Scrum con Ángel Medinilla
- » Resumen de la cuarta charla gratuita de Autentia: SCRUM (con video)

+Comentarios Cómic

+Enlaces

Catálogo de servicios Autentia (PDF 6,2MB)



En formato comic...

Tutorial desarrollado por

Borja Lázaro de Rafael

Consultor tecnológico de desarrollo de proyectos informáticos.

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.



[Catálogo de cursos](#)

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

Fecha de creación del tutorial: 2009-07-08

AOP con AspectJ y Maven

Índice de contenido

- [Introducción.](#)
- [Entorno.](#)
- [AOP \(Aspect Oriented Programming\).](#)
- [Integración de AspectJ con Maven.](#)
- [Ejemplo](#)
- [Probando el ejemplo](#)
- [Conclusiones](#)

Introducción.

La programación orientada a aspectos es algo que cada vez está más presente en varios proyectos, y que gracias a [Spring AOP](#) se integra perfectamente en el framework de Spring como podemos ver en el tutorial [Spring AOP: Cacheando aplicaciones usando anotaciones y aspectos con Aspectj](#) de nuestro compañero Carlos. Pero no en todos los proyectos contamos con [Spring](#), pero no por esto tenemos que descartar la programación de aspectos si es que nos hacen falta.

En este tutorial vamos a ver un ejemplo de programación de aspectos con [AspectJ](#) y como podemos integrarla dentro de nuestros proyectos de [Maven](#) mediante un plugin de compilación, sin necesidad de integrarlo con el framework de Spring.

Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portatil Samsung R70 (Intel(R) Core(TM)2 Duo 2,5Ghz, 2046 MB RAM, 232 Gb HD)
- Sistema Operativo: Windows Vista Home Premium
- Máquina Virtual Java: JDK 1.5.0_14 de Sun Microsystems (http://java.sun.com/javase/downloads/index_jdk5.jsp)
- IDE Eclipse 3.4 (Ganymede) (<http://www.eclipse.org/downloads/>)

AOP (Aspect Oriented Programming).

En este punto vamos a presentar los conceptos básicos de AOP que sirvan de base para alguien nuevo o de refresco para aquellos que lo conocen.

De forma general podemos decir que la programación orientada a aspectos trata de identificar y ocuparse de aquellos comportamientos que afectan de forma transversal a nuestra lógica de negocio. Típicamente son del tipo:

- Comprobaciones de seguridad (verificación de credenciales).
- Gestión de transacciones (apertura y cierre)
- Volcado de trazas.
- etc.



Web

www.adictosaltrabajo.com

Últimos tutoriales

2009-07-08
[AOP con AspectJ y Maven](#)

2009-07-07
[Instalación y configuración de Eclipse Galileo](#)

2009-07-07
[Iniciarse en el manejo de JME, Creación de un Cloth.](#)

2009-07-06
[Primeros pasos con Blender](#)

2009-07-06
[DBUnit-Exportar e Importar BBDD](#)

2009-07-05
[JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones](#)

2009-07-02
[Axis2: Invocación de Servicios Web usando distintos MEP](#)

2009-07-02
[Instalación OpenOffice](#)

2009-07-02
[Juegos 3D en Java: Blender y JMonkeyEngine](#)

2009-06-20
[StAX \(Xml Pull Parser\): Streaming API para XML](#)

2009-06-15
[Configuración de la desconexión de usuarios con ICEFaces](#)

Con AOP podemos implementar este tipo de comportamientos sin necesidad de modificar el código propio de la lógica de negocio.
Para terminar de asentar las bases de AOP vamos a ver un poco de su terminología principal:

- **Aspect:** Es el comportamiento transversal que se aplica sobre aquella funcionalidad configurada bajo el aspecto. (verificación de credenciales, transacciones, trazas, etc.)
- **Join point:** Es cualquier punto de la aplicación sobre el que se puede aplicar el aspecto. (Llamadas a métodos, acceso a atributos, etc.)
- **Advice:** Es la acción que realiza un aspecto en un determinado *join point*. (volcado de trazas, etc.)
- **Pointcut:** Es un conjunto de *join points* al que aplicar un *advice*. (ej. Todos lo métodos "setXXX" de una clase)

Integración de AspectJ con Maven.

Como hemos avanzado en la introducción, al no tener Spring debemos encontrar un modo de que nuestros aspectos se "enganchen" correctamente a nuestra aplicación, y se ejecuten en los casos necesarios.

AspectJ consigue realizar esto al compilar nuestro proyecto con un compilador propio. Podemos decir que al compilar una clase afectada por algún aspecto, el compilador de AspectJ modifica el código introduciendo las llamadas al aspecto.

Al tener nuestro proyecto gestionado con Maven, es necesario un plugin de Maven que en el momento de compilar realice estas modificaciones. El plugin en concreto es [AspectJ compiler Maven Plugin](#). También deberemos tener incluidas las dependencias propias de AspectJ, por lo que en nuestro fichero "pom.xml" deberemo incluir el siguiente código:

```
view plain print ?
01. <project xmlns="http://maven.apache.org/POM/4.0.0"
02.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03.     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
04.     ....
05.     <build>
06.     <plugins>
07.     ....
08.     <plugin>
09.         <groupId>org.codehaus.mojo</groupId>
10.         <artifactId>aspectj-maven-plugin</artifactId>
11.         <version>1.1</version>
12.         <configuration>
13.             <source>1.5</source>
14.         </configuration>
15.         <executions>
16.             <execution>
17.                 <goals>
18.                     <goal>compile</goal>           <!-- use this goal to weave all your main classe
19.                     <goal>test-compile</goal>      <!-- use this goal to weave all your test classe
20.                 </goals>
21.             </execution>
22.         </executions>
23.     </plugin>
24.     ....
25. </plugins>
26. </build>
27.     ....
28.     <dependencies>
29.     ....
30.     <!-- AspectJ -->
31.     <dependency>
32.         <groupId>org.aspectj</groupId>
33.         <artifactId>aspectjrt</artifactId>
34.         <version>1.6.2</version>
35.     </dependency>
36.     ....
37. </dependencies>
38.     ....
39. </project>
```

De la configuración de este plugin, lo más destacable es que debe ejecutarse en el momento de compilación tal y como podemos ver en sus "goals" específicos. Pero otro dato importante es su configuración con el elemento "<source>" con valor "1.5"; que nos sirve para indicar que el nivel de compilación se corresponde con Java 1.5, de esta forma podemos utilizar anotaciones para nuestros aspectos, algo que nos simplifica bastante la configuración de los mismos.

Ejemplo.

Para ver rápidamente como podemos crear aspectos, vamos a implementar un sistema de trazas con aspectos. Tenemos el típico CRUD (Create, Read, Update, Delete) de contactos, donde deseamos que cada vez que se inserte, modifique o borre un contacto se deje una traza con el tipo de acceso.

Primero nos creamos nuestras clases de trazas.

2009-06-10
[LWUIT: Una librería gráfica tipo AWT o Swing para J2ME](#)

2009-06-10
[Mapas mentales con XMind](#)

2009-02-26
[Redimensionar Imagenes en Windows Vista](#)

2009-06-08
[UploadFile con Icefaces + Hibernate + Anotaciones](#)

2009-06-05
[Habilitar exportación en Liferay](#)

2009-06-01
[Registrar Liferay en Eclipse](#)

2009-05-29
[Liferay Social Office](#)

2009-05-28
[Broadcast con Ustream](#)

2009-05-25
[Tabla datos accesible con ordenación y paginación](#)

2009-05-21
[Primeros pasos con Audacity: Un editor de sonido libre y multiplataforma.](#)

2009-05-11
[Introducción a TortoiseSVN](#)

2009-05-07
[Hacer 'scp' de varios ficheros sin solicitud de clave](#)

2009-05-02
[Plugin Hibernate3 para Maven](#)

2009-04-26
[AgileDraw: una técnica rápida de modelado](#)

2009-04-24
[Spring AOP: Cacheando aplicaciones usando anotaciones y aspectos con Aspectj](#)

2009-04-20
[Modelos de conocimiento con CmapTools](#)

2009-04-16
[Informes Crosstab con iReport](#)

2009-04-16
[Registro de un fichero de datos personales con el formulario NOTA](#)

2009-04-15
[Estadísticas de www.adictosaltrabajo.com](#)

```

01. package com.autentia.tutoriales.aspectj.businessobject;
02.
03. import java.io.Serializable;
04. import java.util.Date;
05.
06. import javax.persistence.Column;
07. import javax.persistence.Entity;
08. import javax.persistence.GeneratedValue;
09. import javax.persistence.Id;
10. import javax.persistence.Inheritance;
11. import javax.persistence.InheritanceType;
12. import javax.persistence.Lob;
13. import javax.persistence.Temporal;
14. import javax.persistence.TemporalType;
15.
16. import org.apache.commons.lang.builder.EqualsBuilder;
17. import org.apache.commons.lang.builder.HashCodeBuilder;
18.
19. @Entity
20. @Inheritance(strategy = InheritanceType.JOINED)
21. public abstract class Trace implements Serializable {
22.
23.     private static final long serialVersionUID = -3089626883233965972L;
24.
25.     private Integer id;
26.
27.     private Date whenDate = new Date();
28.
29.     private String description;
30.
31.     @SuppressWarnings("unused")
32.     private Type type;
33.
34.
35.     @Id
36.     @GeneratedValue
37.     public Integer getId() {
38.         return id;
39.     }
40.
41.     @SuppressWarnings("unused")
42.     private void setId(Integer id) {
43.         this.id = id;
44.     }
45.
46.     @Lob
47.     @Column(nullable = false)
48.     public String getDescription() {
49.         return description;
50.     }
51.
52.     @Column(nullable = false)
53.     @Temporal(TemporalType.TIMESTAMP)
54.     public Date getWhenDate() {
55.         return whenDate;
56.     }
57.
58.     @SuppressWarnings("unused")
59.     private void setWhenDate(Date whenDate) {
60.         this.whenDate = whenDate;
61.     }
62.
63.     public void setDescription(String description) {
64.         this.description = description;
65.     }
66.
67.
68.     @Override
69.     public int hashCode() {
70.         final HashCodeBuilder hashCodeBuilder = new HashCodeBuilder();
71.         hashCodeBuilder.append(getWhenDate());
72.         hashCodeBuilder.append(getDescription());
73.         return hashCodeBuilder.toHashCode();
74.     }
75.
76.     public boolean equals(Object obj) {
77.         boolean isEqual = false;
78.         try {
79.             if (this == obj)
80.                 return true;
81.             final Trace other = (Trace) obj;
82.             final EqualsBuilder eqb = new EqualsBuilder();
83.             eqb.append(getWhenDate(), other.getWhenDate());
84.             eqb.append(getDescription(), other.getDescription());
85.             isEqual = eqb.isEquals();
86.         } catch (Exception e) {
87.             isEqual = false;
88.         }
89.         return isEqual;
90.     }
91.
92.     public abstract Type getType();
93.
94.     @SuppressWarnings("unused")
95.     private void setType(Type type) {
96.         this.type = type;
97.     }
98.
99.     public enum Type {INSERT, UPDATE, DELETE}

```

Abril 2009

2009-04-15

[Iniciación a OSWorkflow con Spring](#)

2009-04-14

[Tests de Selenium con librerías de componentes JSF: Apache Tomahawk.](#)

2009-04-13

[JTAPI. El API de Telefonía para Java](#)

2009-04-13

[Registro de Web Services con Apache jUDDI. Configuración y ejemplo](#)

2009-04-13

[Cómo hacer UML con Eclipse y el plugin UML2](#)

2009-04-09

[Spring WS: Servicios Web a través del correo electrónico](#)

2009-04-02

[Creación de cursos con Moodle](#)

2009-03-31

[Integrar Liferay Portal 5.2.1 con Pentaho BI 2.0.0 sobre MySQL 5.1](#)

2009-03-31

[Spring WS: Construcción de Clientes de Servicios Web con Spring](#)

2009-03-30

[Administración de sitios Moodle](#)

2009-03-29

[Empaquetamiento de aplicaciones de escritorio \(standalone\) con Maven](#)

2009-03-27

[Primeros pasos con Moodle](#)

2009-03-26

[Introducción a JSF Java](#)

2009-03-25

[A1 Website Analyzer](#)

2009-03-24

[Cómo ver el correo de Gmail sin conexión a Internet](#)

2009-03-20

[JasperReports Maven Plugin](#)

2009-03-16

[Creación de contenidos SCORM: eXe](#)

2009-03-15

[Spring WS: Creación de](#)

```

view plain print ?
01. package com.autentia.tutoriales.aspectj.businessobject;
02.
03. import javax.persistence.Entity;
04.
05. @Entity
06. public class InsertTrace extends Trace {
07.
08.     /**
09.      *
10.      */
11.     private static final long serialVersionUID = 6876829701547179312L;
12.
13.     @Override
14.     public Type getType() {
15.         return Type.INSERT;
16.     }
17.
18. }

```

2009-03-13
[Instalación Alfresco \(Labs\)](#)

2009-02-26
[Maven JXR Plugin: publica el código fuente en el site](#)

Últimas ofertas de empleo

2009-06-29
[Atención a cliente - Call Center - BARCELONA.](#)

2009-06-25
[Atención a cliente - Call Center - BARCELONA.](#)

2009-06-20
[Comercial - Ventas - CASTELLON.](#)

2009-06-19
[Otras - Ingeniería \(minas, puentes y puertos\) - VALENCIA.](#)

2009-06-17
[Comercial - Ventas - ALICANTE.](#)

```

view plain print ?
01. package com.autentia.tutoriales.aspectj.businessobject;
02.
03. import javax.persistence.Entity;
04.
05. /**
06.  *
07.  *
08.  */
09. @Entity
10. public class UpdateTrace extends Trace {
11.
12.
13.     private static final long serialVersionUID = 47615098436209786L;
14.
15.     @Override
16.     public Type getType() {
17.         return Type.UPDATE;
18.     }
19.
20. }

```

Anuncios Google

[RPC Plugin](#)

[Pesticide Package](#)

[LCD Monitor Import](#)

[Descargar Plugin](#)

```

view plain print ?
01. package com.autentia.tutoriales.aspectj.businessobject;
02.
03. import javax.persistence.Entity;
04.
05. /**
06.  *
07.  *
08.  */
09. @Entity
10. public class DeleteTrace extends Trace {
11.
12.
13.     private static final long serialVersionUID = 47615098436209786L;
14.
15.     @Override
16.     public Type getType() {
17.         return Type.DELETE;
18.     }
19.
20. }

```

Ahora debemos crearnos el DAO de acceso a datos. (Con ver el interfaz el suficiente para nuestro ejemplo, ahora veréis)

```

view plain print ?
01. package com.autentia.tutoriales.aspectj.dao;
02.
03. import java.util.List;
04.
05. import com.autentia.tutoriales.aspectj.TraceAnnotation;
06. import com.autentia.tutoriales.aspectj.businessobject.Trace.Type;
07.
08. public interface Dao {
09.
10.     @TraceAnnotation(type=Type.DELETE)
11.     void delete(Object entity);
12.
13.     <T> List<T> loadAll(Class<T> entityClass);
14.
15.     @TraceAnnotation(type=Type.INSERT)
16.     void insert(Object entity);
17.
18.     @TraceAnnotation(type=Type.UPDATE)
19.     <T> T update(T entity);
20.
21.
22. }

```

El punto interesante, se encuentra en que los métodos "delete", "insert" y "update" se encuentran anotado bajo la anotación "@TraceAnnotation". Esta anotación es la que nos sirve para indicar que éstos métodos serán "join points" de nuestro aspecto.

La anotación la hemos creado para marcar los métodos bajo el aspecto que vamos a crear, por lo que hemos creado una anotación específica para métodos y su código es el siguiente:

```
01. package com.autentia.tutoriales.aspectj;
02.
03. import java.lang.annotation.Retention;
04. import java.lang.annotation.RetentionPolicy;
05. import java.lang.annotation.Target;
06. import java.lang.annotation.ElementType;
07.
08. import com.autentia.tutoriales.aspectj.businessobject.Trace.Type;
09.
10. @Retention(RetentionPolicy.RUNTIME)
11. @Target(ElementType.METHOD)
12. public @interface TraceAnnotation {
13.     public Type type() default Type.INSERT;
14.
15. }
```

Finalmente nos queda por crear nuestro aspecto, que será el responsable de registrar los distintos tipos de trazas. Su código es:

```

01. package com.autentia.tutoriales.aspectj;
02.
03. import java.lang.reflect.Method;
04.
05. import org.apache.commons.logging.Log;
06. import org.apache.commons.logging.LogFactory;
07. import org.aspectj.lang.JoinPoint;
08. import org.aspectj.lang.annotation.After;
09. import org.aspectj.lang.annotation.Aspect;
10. import org.aspectj.lang.reflect.MethodSignature;
11.
12. import com.autentia.tutoriales.aspectj.businessobject.DeleteTrace;
13. import com.autentia.tutoriales.aspectj.businessobject.InsertTrace;
14. import com.autentia.tutoriales.aspectj.businessobject.Trace;
15. import com.autentia.tutoriales.aspectj.businessobject.UpdateTrace;
16. import com.autentia.tutoriales.aspectj.businessobject.Trace.Type;
17. import com.autentia.tutoriales.aspectj.dao.Dao;
18. import com.autentia.tutoriales.aspectj.dao.DaoFactory;
19.
20. @Aspect
21. public class TraceAspect {
22.
23.     /** Logger */
24.     private static final Log log = LogFactory.getLog(TraceAspect.class);
25.
26.     private final static String TYPE_KEY = "type";
27.
28.     @After("@annotation(com.autentia.tutoriales.aspectj.TraceAnnotation)")
29.     public void addTrace(JoinPoint call) {
30.         try {
31.             Object entity = call.getArgs()[0];
32.             if (!(entity instanceof Trace)) {
33.                 Type type = this.getAnnotationType(call);
34.                 Trace trace = null;
35.                 if (type.equals(Type.INSERT)) {
36.                     trace = new InsertTrace();
37.                 } else if (type.equals(Type.UPDATE)) {
38.                     trace = new UpdateTrace();
39.                 } else if (type.equals(Type.DELETE)) {
40.                     trace = new DeleteTrace();
41.                 }
42.                 if (trace != null) {
43.                     trace.setDescription(entity.toString());
44.                     Dao dao = DaoFactory.getDao();
45.                     dao.insert(trace);
46.                 }
47.             }
48.         } catch (SecurityException e) {
49.             log.error(e);
50.         } catch (NoSuchMethodException e) {
51.             log.error(e);
52.         }
53.     }
54.
55.
56.
57.     /**
58.      * @return Devuelve el tipo de la anotación
59.      * @throws NoSuchMethodException
60.      * @throws SecurityException
61.      * @throws ClassNotFoundException
62.      */
63.     private Type getAnnotationType(JoinPoint call) throws SecurityException,
64.         NoSuchMethodException {
65.         Method metodo = this.getCallMethod(call);
66.
67.         TraceAnnotation anotacion = metodo.getAnnotation(TraceAnnotation.class);
68.         return anotacion.type();
69.     }
70.
71.
72.     /**
73.      * Returns the intercepted method
74.      *
75.      * @param call
76.      * @return
77.      */
78.     private Method getCallMethod(JoinPoint call) {
79.         Method metodo = null;
80.         MethodSignature sig = (MethodSignature) call.getSignature();
81.         metodo = sig.getMethod();
82.
83.         return metodo;
84.     }
85. }

```

Aquí lo más importante es ver como la clase está anotada bajo la anotación "@Aspect", que es una anotación propia de AspectJ, y que sirve para indicarle al compilador de AspectJ que esta clase es un aspecto.

Otro punto a destacar es el método "addTrace", que es el *advice* del aspecto (comportamiento a ejecutar). Se puede observar cómo este método está anotado con la anotación "@After" que indica que debe ejecutarse después de la llamada a los métodos anotados con "@TraceAnnotation".

Otras posibles opciones sería "@Before", para indicar que se ejecute antes de la llamada al método y "@Around" para indicar que debe ejecutarse tanto antes de la llamada al método como después.

Si observamos el código podemos ver como lo primero que hacemos es comprobar que la entidad sobre la que se está realizando la operación (insert, update o delete) no es de tipo "Trace" ya que ésta no interesan. Después recuperamos el tipo de operación,

creamos la traza correspondiente y la guardamos en la base de datos.

Probando el ejemplo

Para poder probar nuestro aspecto nos creamos un test unitario que inserte, modifique y borre un contacto y después de cada acción comprobaremos que se ha insertado la traza. El código del test es:

```
view plain print ?
01. package com.autentia.tutoriales.aspectj.test;
02.
03. import junit.framework.Assert;
04.
05. import org.junit.AfterClass;
06. import org.junit.BeforeClass;
07. import org.junit.Test;
08.
09. import com.autentia.tutoriales.aspectj.businessobject.Contact;
10. import com.autentia.tutoriales.aspectj.businessobject.Trace;
11. import com.autentia.tutoriales.aspectj.dao.Dao;
12. import com.autentia.tutoriales.aspectj.dao.DaoFactory;
13.
14. public class DaoImplTest {
15.
16.     private static Dao dao;
17.
18.     @BeforeClass
19.     public static void init() {
20.
21.         DaoFactory.init();
22.         dao = DaoFactory.getDao();
23.
24.
25.         DaoFactory.openSession();
26.
27.     }
28.
29.     @AfterClass
30.     public static void end() {
31.         DaoFactory.closeSession();
32.         DaoFactory.close();
33.     }
34.
35.     @Test
36.     public void myTest() {
37.
38.         Contact contact = new Contact();
39.         contact.setName("Borja");
40.         contact.setSurname("Lázaro");
41.         contact.setEmail("blazaro@autentia.com");
42.
43.         dao.insert(contact);
44.
45.         Assert.assertEquals(1, dao.loadAll(Contact.class).size());
46.         Assert.assertEquals(1, dao.loadAll(Trace.class).size());
47.
48.         contact.setName("Borja modificado");
49.
50.         dao.update(contact);
51.
52.         Assert.assertEquals(1, dao.loadAll(Contact.class).size());
53.         Assert.assertEquals(2, dao.loadAll(Trace.class).size());
54.
55.         dao.delete(contact);
56.
57.         Assert.assertEquals(0, dao.loadAll(Contact.class).size());
58.         Assert.assertEquals(3, dao.loadAll(Trace.class).size());
59.
60.     }
61. }
```

Conclusiones

Finalmente podemos ver cómo con AOP podemos implementar aquella funcionalidad que es transversal al resto de la aplicación, sin necesidad de tener que cambiar el código propio de la lógica de negocio. También se puede observar como se consigue una mayor independencia de este tipo de tareas respecto al resto de la aplicación, el famoso bajo acoplamiento de las aplicaciones. Por ejemplo, si sobre el caso mostrado se quisiera cambiar el sistema de trazas a otro distinto, sólo habría que cambiar la implementación del aspecto, dejando el resto del código de nuestra aplicación sin tocar.

Si queréis, aquí podéis conseguir todo el código fuente de este ejemplo [AOP con AspectJ y Maven](#).

Un saludo.

Borja Lázaro de Rafael.

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

Muy malo Malo Regular Bueno Muy bueno



Votar

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.

Nombre:

E-Mail:

Comentario:

Enviar comentario

[Texto Legal y condiciones de uso](#)

- 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. 



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
DBUnit-Exportar e Importar BBDD	DBUnit como complemento de los test unitarios	2009-07-06	218	Muy bueno	1	
JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones	En este tutorial Carlos García nos enseñará a grabar y reproducir navegaciones con JMeter, para poder realizar pruebas de carga o stress sobre aplicaciones Web	2009-07-05	291	Regular	2	
Configuración de la desconexión de usuarios con ICEFaces	Este tutorial muestra la manera de configurar y traducir la ventana de desconexión o pérdida de sesión del usuario en ICEFaces.	2009-06-15	698	Muy bueno	7	
Tabla datos accesible con ordenación y paginación	En este tutorial vamos a ver como podemos hacer una tabla de datos con ordenación y paginación aplicando los criterios de accesibilidad y los conceptos de mejora progresiva y Javascript no obstrusivo.	2009-05-25	1604	Muy bueno	12	
Plugin Hibernate3 para Maven	En este tutorial veremos las posibilidades que nos ofrece el plugin de Hibernate3 para Maven, como por ejemplo, la generación del esquema de base de datos desde clases con anotaciones.	2009-05-02	1013	Bueno	7	
Creación de cursos con Moodle	En este tercer tutorial de la serie veremos en detalle la creación de un curso	2009-04-02	2770	Muy bueno	11	
Empaquetamiento de aplicaciones de escritorio (standalone) con Maven	En este tutorial vamos a aprender a empaquetar aplicaciones de escritorio con Maven para ser distribuidas	2009-03-29	1851	Muy bueno	14	
Cómo ver el correo de Gmail sin conexión a Internet	En este tutorial vamos a ver como podemos configurar el navegador Firefox 3 para poder acceder a todo nuestro histórico de correos sin necesitar de un conexión a Internet.	2009-03-24	2415	Bueno	12	
JasperReports Maven Plugin	JasperReports Maven Plugin es un plugin de Maven que nos permite compilar nuestros informes JasperReports.	2009-03-20	2274	Bueno	10	
Maven JXR Plugin: publica el código fuente en el site	JXR es un plugin para maven, de tipo reporting, que genera en el site un informe con los fuentes java de tu aplicación: "Source Xref".	2009-02-26	1504	Muy bueno	5	

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.