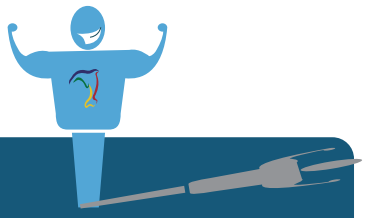
autentia

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...

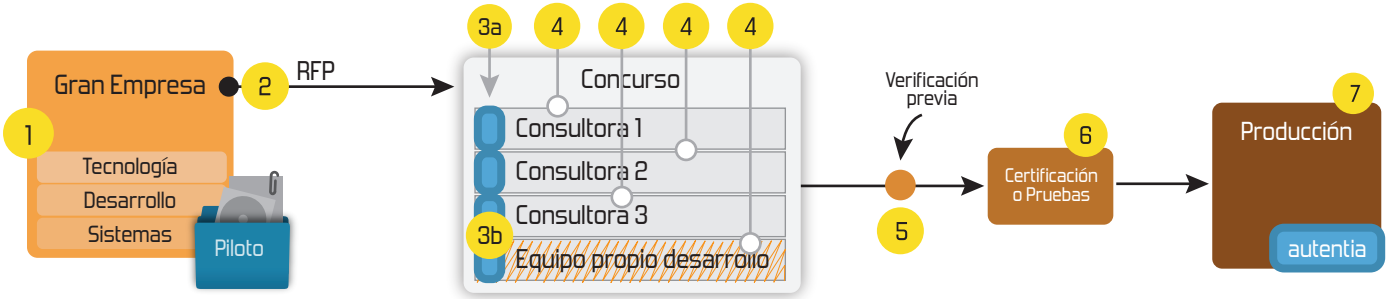
1. Desarrollo de componentes y proyectos a medida



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

Últimos tutoriales

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 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

Tutorial desarrollado por

Carlos García Pérez

Creador de MobileTest, un complemento educativo para los profesores y sus alumnos.

Consultor tecnológico en el desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática *

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Descargar este documento en formato PDF: spring_aop_cache_aspecj.pdf

Fecha de creación del tutorial: 2009-04-24

Spring AOP: Cacheando aplicaciones usando anotaciones y aspectos con Aspectj

Introducción.

Para empezar este tutorial, os invito a observar la siguiente clase Java poniendo especial atención a los métodos `getAll` y `add`.

view plain print ?

```
01. package com.autentia.tutoriales.spring.aop.aspectj;
02.
03. import com.autentia.tutoriales.spring.aop.aspectj.cache.Cachea;
04. import com.autentia.tutoriales.spring.aop.aspectj.cache.Descachea;
05.
06. /**
07.  * Bean de nuestra aplicación con métodos que cachean y descachean
08.  * @author Carlos García. Autentia.
09.  * @see http://www.mobiletest.es
10.  */
11. public class Provincias  {
12.     private java.util.List<String> provincias;
13.
14.     /**
15.      * Constructor
16.      */
17.     public Provincias(){
18.         this.provincias = new java.util.ArrayList<String>();
19.     }
20.
21.     /**
22.      *(SIMULAMOS UNA OPERACION COSTOSA CACHEAMOS EL RESULTADO CON TIEMPO INFINITO)
23.      * @return Devuelve todas las provincias
24.      */
25.     @Cachea(cacheKey="Provincias.provincias", expireTime=0)
26.     public java.util.List<String> getAll(){
27.         try {
28.             Thread.sleep(2000);
29.         } catch (InterruptedException e) {}
30.
31.         return provincias;
32.     }
33.
34.     /**
35.      * Añadimos una provincia (LIMPIAMOS LA CACHE)
36.      * @param provincia Provincia a añadir
37.      */
38.     @Descachea(cacheKey="Provincias.provincias")
39.     public void add(String provincia){
40.         provincias.add(provincia);
41.     }
42. }
```

Ahora bien al igual que pasa en la mayoría de las aplicaciones, hay muchos métodos que siempre devuelven lo mismo (getAll) a no ser que otro método (add) invalide el mismo.
Estos métodos son claros candidatos de ser cacheados, pues ¿para qué volver a consultar a la BD y traernos los resultados por la red?, ¿para que volver a invocar un servicio web si estamos seguros de que el resultado será el mismo?. ¿Para qué....?

Pues bien, en este tutorial vamos a usar la **programación orientada a aspectos** para dotar con un par de simples anotaciones los métodos cuyo resultado queremos cachear y los métodos que invalidan la cache(s) establecidas.

No os voy a tratar temas teóricos sobre programación orientada a Aspectos, Spring, Maven... para eso están los libros, Internet o los cursos que impartimos en Autentia, sólo os quiero presentar **un completo ejemplo práctico de creación de un aspecto basado en Aspectj y**

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 Servicios Web con Spring

2009-03-13
Instalación Alfresco (Labs)

2009-02-26
Maven JXR Plugin: publica el código fuente en el site

2009-03-15
Generación de XML Schema (XSD) y DTD a partir de documentos XML

2009-03-04
Persistencia con Spring

2009-02-26
Vistas materializadas

2009-02-03
Instalación de MySQL 5.1 en Windows

2009-03-03
Instalación de Java Virtual Machine

2009-03-03
Primeros Pasos con Liferay 5.2.1

2009-02-27
Edicion de video MPEG2

2009-02-26
Introducción teórica a XPath

2009-02-26
Integración Selenium / Maven 2 / Surefire / Cargo / Tomcat 6

2009-02-24
Selenium Remote Control

2009-02-22
Integración de Groovy, JRuby y BeanShell con Spring 2

2009-02-18
Instalación de Pentaho BI Suite

anotaciones.

Más adelante escribiremos un test con JUnit que invoque los métodos `getAll` y `add`, observe cual será la salida de la aplicación:

```
...........
Llamada add para forzar el descacheo
Llamada getAll: 2010 milisegundos
Llamada getAll: 0 milisegundos
Llamada add para forzar el descacheo
Llamada getAll: 2002 milisegundos
FIN
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.969 sec
...........
```

Observe lo siguiente:

- En la segunda invocación del método `getAll` nos ahorramos unos segundos de ejecución y recursos.. que hoy en día las aplicaciones a veces van más lentas que hace 8 años usando en la actualidad máquinas más potentes ;-)
- Posteriormente, una vez descacheada la información, el tiempo de ejecución de `getAll` vuelve a incrementarse, es decir, la caché fue limpiada.

Manos a la obra con el ejemplo:

Antes de nada, como verá en la sección referencias, hay muchas formas de hacer lo mismo, pero desde mi punto de vista esta forma que he diseñado gana en sencillez, bajo acoplamiento y esfuerzo para transladarlo a vuestros proyectos.

Os dejo el código fuente del proyecto (proyecto Eclipse, Maven) para que realiceis vuestras pruebas.

Creamos una anotación para cachear el resultado de cualquier método de nuestras aplicaciones:

view plain print ?

```
01. package com.autentia.tutoriales.spring.aop.aspectj.cache;
02.
03. import java.lang.annotation.*;
04.
05. /**
06.  * Anotación para métodos que realizen operaciones de cacheo de información
07.  * @author Carlos García. Autentia.
08.  */
09. @Retention(RetentionPolicy.RUNTIME)
10. @Target(ElementType.METHOD)
11. public @interface Cachea {
12.     /**
13.       * @return Identificador dentro de la cache con la que se guardará la información
14.       */
15.     public String cacheKey();
16.
17.     /**
18.       * @return Número de milisegundos que tendrá validez la cache
19.       * (0=infinito la información se guarda en disco su cerramos la aplicación o cuando se necesita "pagina
20.       */
21.     public int expireTime();
22. }
```

Creamos una anotación para limpiar la cache creada usando la anotación anterior:

view plain print ?

```
01. package com.autentia.tutoriales.spring.aop.aspectj.cache;
02.
03. import java.lang.annotation.*;
04.
05. /**
06.  * Anotación para métodos que realizen operaciones de descacheo de información
07.  * @author Carlos García. Autentia.
08.  */
09. @Retention(RetentionPolicy.RUNTIME)
10. @Target(ElementType.METHOD)
11. public @interface Descachea {
12.     /**
13.       * @return Identificador de la información cacheada a descachear
14.       */
15.     public String cacheKey();
16. }
```

Aspecto que automáticamente será invocado en los métodos que estén anotados para ser cacheados o descacheados:

Community Edition 1.7.0

2009-02-18
Replicar Web PHP en máquina local

2009-02-16
Selenium Core : El motor de Selenium.

2009-02-16
Integración de JasperReports con PHP

2009-02-09
EJB 3.0 y pruebas unitarias con Maven, JUnit 4 y Embedded JBoss sobre Java 6

2009-02-09
Web Service Security

2009-02-09
Manual Avanzado de Firebug

2009-01-29
Ejemplo con Mockito

2009-01-29
Uso de Mock objects en pruebas con Mockito

2009-01-29
StrutsTestCase

2009-01-28
Eventos en Hibernate (parte III)

2009-01-28
Eventos en Hibernate (parte II)

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

2009-01-25
Aprendiendo XMLSchema a través de ejemplos

view plain print ?

```
01. package com.autentia.tutoriales.spring.aop.aspectj.cache;
02.
03. import java.lang.reflect.Method;
04. import com.opensymphony.oscache.general.GeneralCacheAdministrator;
05. import com.opensymphony.oscache.web.filter.ExpiresRefreshPolicy;
06. import com.opensymphony.oscache.base.NeedsRefreshException;
07. import java.util.Properties;
08. import org.aspectj.lang.Signature;
09.
10. /**
11.  * Aspecto que se encarga del sistema de cache de la aplicación
12.  * @author Carlos García. Autentia.
13.  * @see http://www.mobiletest.es
14.  */
15. @org.aspectj.lang.annotation.Aspect
16. public class CacheAspect {
17.
18.     private final int    NO_EXPIRE   = 0;
19.     private final String CACHE_KEY   = "cacheKey";
20.     private final String EXPIRE_TIME = "expireTime";
21.
22.     private GeneralCacheAdministrator cache;
23.
24.     /**
25.      * Sí, si.. está acoplado a OSCache, podría haber definido una inferfaz que me desacoplara del sistema
26.      * Recuerda que esto es un tutorial y no tenia ganas.
27.      * @param cache Sistema de cache nativo, no reinventemos la rueda :-)
28.      */
29.     public CacheAspect(com.opensymphony.oscache.general.GeneralCacheAdministrator cache){
30.         this.cache = cache;
31.     }
32.
33.     /**
34.      * Este método será llamado por AOP en tiempo de ejecución para aquellos
35.      * métodos con la anotación Cachea
36.      */
37.     @org.aspectj.lang.annotation.Around("@annotation(Cachea)")
38.     public Object cachear(org.aspectj.lang.ProceedingJoinPoint call) throws Throwable {
39.         Properties cacheProps = this.getAnnotationProperties(call, true);
40.         Object   result       = null;
41.         String   cacheKey     = cacheProps.getProperty(CACHE_KEY);
42.         int      expire       = NO_EXPIRE;
43.
44.         expire   = Integer.parseInt(cacheProps.getProperty(EXPIRE_TIME));
45.         // Evitamos números negativos como valor de tiempo de validez de la cache
46.         if (expire < 0){
47.             expire = NO_EXPIRE;
48.         }
49.
50.         try {
51.              result  = cache.getFromCache(cacheKey);
52.         } catch (NeedsRefreshException e) {
53.              // Los datos de la cache no existen o han caducado
54.             cache.cancelUpdate(cacheKey);
55.         }
56.
57.         if (result == null) {
58.             // Actualmente no hay datos cacheados validos, ejecutamos el método y
59.             // cacheamos el resultado.
60.             result = call.proceed();
61.
62.             if (expire == 0){
63.                 // No se especificó un tiempo de validez
64.                 cache.putInCache(cacheKey, result);
65.             } else {
66.                 // Tiene un tiempo de validez
67.                 cache.putInCache(cacheKey, result, new ExpiresRefreshPolicy(expire));
68.             }
69.         }
70.
71.         return result;
72.     }
73.
74.     /**
75.      * Este método será llamado por AOP en tiempo de ejecución.
76.      * Vemos si el método tiene parámetros de descacheo a través de la anotación Descachea
77.      */
78.     @org.aspectj.lang.annotation.Around("@annotation(Descachea)")
79.     public Object descachear(org.aspectj.lang.ProceedingJoinPoint call) throws Throwable {
80.         // Ejecutamos el método
81.         Object result    = call.proceed();
82.
83.         // Descacheamos
84.         Properties cacheProps = this.getAnnotationProperties(call, false);
85.         String    cacheKey    = cacheProps.getProperty(CACHE_KEY);
86.         cache.removeEntry(cacheKey);
87.
88.         // Devolvemos el resultado del método
89.         return result;
90.     }
91.
92.
93.     /**
94.      * @return Devuelve un properties con los atributos de la anotación (Cachea o Descachea)
95.      */
96.     private Properties getAnnotationProperties(org.aspectj.lang.ProceedingJoinPoint call, boolean isCacheo) {
97.         Properties  properties = new Properties();
98.         Method      metodo     = this.getCallMethod(call);
99.
100.        if (isCacheo){
101.            Cachea  anotacion  = metodo.getAnnotation(Cachea.class);
102.            properties.put(CACHE_KEY, anotacion.cacheKey());
103.            properties.put(EXPIRE_TIME, String.valueOf(anotacion.expireTime()));
104.        } else {
105.            Descachea anotacion = metodo.getAnnotation(Descachea.class);
106.            properties.put(CACHE_KEY, anotacion.cacheKey());
107.        }
```

Archivo de configuración de Spring 2 `(/main/resources/applicationContext.xml)`**:**

view plain print ?

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <beans xmlns="http://www.springframework.org/schema/beans"
03.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04.     xmlns:aop="http://www.springframework.org/schema/aop"
05.     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
/spring-beans-2.5.xsd
06.                         http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
aop-2.5.xsd">
07.
08.     <!--  Bean de la lógiga de negocio de mi aplicación de ejemplo. -->
09.     <bean id="provincias"   class="com.autentia.tutoriales.spring.aop.aspectj.Provincias"/>
10.
11.     <!-- Aspecto que proporciona cacheo/descacheo de información del resto del sistema -->
12.     <bean id="cacheService" class="com.autentia.tutoriales.spring.aop.aspectj.cache.CacheAspect">
13.         <constructor-arg index="0">
14.             <!-- Le inyectamos el sistema de cache nativo a usar. -->
15.             <!-- Sí, si.. está acoplado a OSCache, podría haber definido una inferfaz que me desacoplara del sistema c
16.             <bean class="com.opensymphony.oscache.general.GeneralCacheAdministrator">
17.                 <constructor-arg index="0">
18.                     <props>
19.                         <!-- Habilitamos el cache en disco -->
20.                         <prop key="cache.persistence.class">com.opensymphony.oscache.plugins.diskpersistence.DiskPersi
21.
22.                         <!--  Especificamos el directorio donde se cacheará la información -->
23.                         <prop key="cache.path">${java.io.tmpdir}</prop>
24.
25.                         <!--  Indicamos que queremos un límite también para los elementos cacheados en disco -->
26.                         <prop key="cache.unlimited.disk">false</prop>
27.
28.                         <!--  Como mucho serán cacheados 1000 objetos simultánemente -->
29.                         <prop key="cache.capacity">1000</prop>
30.
31.                         <!--  Cuando se supere el límite de 500 objetos, se aplicará el algoritmo LRU (Least Recently
32.                         espacio en disco y cachear los nuevos elementos. -->
33.                         <prop key="cache.algorithm">com.opensymphony.oscache.base.algorithm.LRUCache</prop>
34.                     </props>
35.                 </constructor-arg>
36.             </bean>
37.         </constructor-arg>
38.     </bean>
39.
40.     <!-- Configura el proxy para los aspectos definidos en nuestra aplicación. -->
41.     <aop:aspectj-autoproxy />
42. </beans>
```

Archivo de configuración de Maven `pom.xml`**:**

view plain print ?

```
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02.   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
03.     <modelVersion>4.0.0</modelVersion>
04.     <groupId>com.autentia.tutoriales</groupId>
05.     <artifactId>spring_aop_cache_aspectj</artifactId>
06.     <packaging>jar</packaging>
07.     <version>1.0-SNAPSHOT</version>
08.     <name>spring_aop_cache_aspectj</name>
09.     <url>http://www.adictosaltrabajo.com</url>
10.     <build>
11.         <plugins>
12.             <!-- Indicamos que use Java 6 -->
13.             <plugin>
14.                 <artifactId>maven-compiler-plugin</artifactId>
15.                 <configuration>
16.                     <source>1.6</source>
17.                     <target>1.6</target>
18.                     <encoding>UTF-8</encoding>
19.                 </configuration>
20.             </plugin>
21.         </plugins>
22.     </build>
23.     <dependencies>
24.         <dependency>
25.             <groupId>org.springframework</groupId>
26.             <artifactId>spring</artifactId>
27.             <version>2.5.6</version>
28.         </dependency>
29.
30.         <dependency>
31.             <groupId>org.springframework</groupId>
32.             <artifactId>spring-core</artifactId>
33.             <version>2.5.6</version>
34.         </dependency>
35.
36.         <dependency>
37.             <groupId>org.springframework</groupId>
38.             <artifactId>spring-aspects</artifactId>
39.             <version>2.5.6</version>
40.         </dependency>
41.
42.         <!-- Cglib AOP Proxy, para Java > 1.4 -->
43.         <dependency>
44.             <groupId>cglib</groupId>
45.             <artifactId>cglib-nodep</artifactId>
46.             <version>2.2</version>
47.         </dependency>
48.
49.         <dependency>
50.             <groupId>opensymphony</groupId>
51.             <artifactId>oscache</artifactId>
52.             <version>2.4</version>
53.         </dependency>
54.
55.         <dependency>
56.             <groupId>junit</groupId>
57.             <artifactId>junit</artifactId>
58.             <version>4.1</version>
59.             <scope>test</scope>
60.         </dependency>
61.     </dependencies>
62. </project>
```

Ejecuto un ejemplo a modo de test funcional:

Puede ser ejecutarlo con la sentencia: `mvn test`.

view plain print ?

```
01. package com.autentia.tutoriales.spring.aop.aspectj.cache;
02.
03. import org.junit.Assert;
04. import org.springframework.context.ApplicationContext;
05. import org.springframework.context.support.ClassPathXmlApplicationContext;
06. import com.autentia.tutoriales.spring.aop.aspectj.Provincias;
07.
08. /**
09.  * Lanzo el ejemplo a modo de test funcional..
10.  * ya se que esto no es un test valido real, sólo lo hago para ahorrarme tiempo
11.  */
12. public class SpringCacheTest  {
13.
14.     private ApplicationContext  factory;
15.
16.     /**
17.       * Inicializamos el contexto de Spring
18.       */
19.     @org.junit.Before
20.     public void initTests(){
21.         this.factory  = new ClassPathXmlApplicationContext("applicationContext.xml");
22.     }
23.
24.     @org.junit.Test
25.     public void testCache() {
26.         Provincias provincias;
27.         java.util.List<String> lista;
28.         long timestamp = 0;
29.
30.         try {
31.             provincias = (Provincias) factory.getBean("provincias") ;
32.
33.             System.err.println("Llamada add para forzar el descacheo");
34.             provincias.add("a");
35.
36.             timestamp  = System.currentTimeMillis();
37.             lista = provincias.getAll();
38.             System.err.println("Llamada getAll: " + (System.currentTimeMillis() - timestamp) + " milisegundos");
39.
40.             timestamp  = System.currentTimeMillis();
41.             lista = provincias.getAll();
42.             System.err.println("Llamada getAll: " + (System.currentTimeMillis() - timestamp) + " milisegundos");
43.
44.             System.err.println("Llamada add para forzar el descacheo");
45.             provincias.add("b");
46.
47.             timestamp  = System.currentTimeMillis();
48.             lista = provincias.getAll();
49.             System.err.println("Llamada getAll: " + (System.currentTimeMillis() - timestamp) + " milisegundos");
50.
51.             System.err.println("FIN");
52.
53.             Assert.assertTrue(true);
54.         } catch (Exception ex){
55.             System.err.println(ex);
56.             Assert.fail();
57.         }
58.     }
59. }
```

Referencias

- http://www.javalobby.org/java/forums/t44746.html
- http://www.proactiva-calidad.com/java/spring/aop2.html
- http://www.oracle.com/technology/pub/articles/dev2arch/2006/05/declarative-caching2.html
- http://opensource.atlassian.com/confluence/spring/display/DISC/Caching+the+result+of+methods+using+Spring+and+EHCache
- http://opensource.atlassian.com/confluence/spring/display/DISC/AOP+Cache
- http://wanghy.sourceforge.net/cache/index.html

Conclusiones

Como habéis podido ver la programación orientada a aspectos deja nuestro código mucho más desacoplado, centrándonos en la lógica de negocio y dejando los temas como seguridad, gestión de trazas, cacheo, etc. al margen del mismo... en este tema Spring nos proporciona un amplísimo abanico de posibilidades.

En Autentia, estamos constantemente formándonos para intentar conseguir cada vez software de más calidad. Espero nos tengais en cuenta si necesitais algún tipo de consultaría o formación a medida.

Al margen de este tutorial, os invito a que profundizeis en esta importante filosofía de desarrollo de sistemas pues como dije antes, esto no es más que un tutorial y no un libro concreto y/o especializado en programación orientada a aspectos, Spring, etc.

Un saludo, espero que os haya parecido útil este tutorial.
Carlos García. Creador de MobileTest, un complemento educativo para los profesores y sus alumnos.

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. ADD THIS BLOG TO MY Technorati FAVORITES

Recuerda

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

¿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

J2EE, EJBs, Struts...

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Iniciación a OSWorkflow con Spring	En este tutorial vamos a presentar uno de los motores de workflow, "OSWorkflow" y su integración con Spring.	2009-04-15	274	Muy bueno	2	
Spring WS: Servicios Web a través del correo electrónico	En este tutorial aprenderemos a configurar un cliente y servicio web para que envie/atienda peticiones por correo electrónico.	2009-04-09	442	Muy bueno	2	
Spring WS: Construcción de Clientes de Servicios Web con Spring	En este tutorial veremos las características que nos ofrece Spring para la contrucción de clientes de servicios Web	2009-03-31	536	Muy bueno	4	
Spring WS: Creación de Servicios Web con Spring	En este tutorial veremos un completo ejemplo de creación de un servicio web Contract-First con Spring y Maven	2009-03-15	1057	Muy bueno	4	
Persistencia con Spring	En el siguiente tutorial vamos a ver algunas de las aportaciones que nos ofrece Spring para mejorar la capa de persistencia de nuestras aplicaciones	2009-03-04	1366	Muy bueno	10	
Integración de Groovy, JRuby y BeanShell con Spring 2	Como Integrar Groovy, JRuby o BeanShell con Spring 2 para realizar comportamientos dinámicos sin recompilar	2009-02-22	623	Muy bueno	1	
OSCache: Sistema de caché para aplicaciones Java	En este tutorial, aprenderemos a usar OSCache como sistema de cache para aplicaciones Java	2009-01-02	1074	Bueno	11	
JMeter: Tests de rendimiento usando varios clientes distribuidos	En este tutorial, aprenderemos a configurar JMeter para realizar pruebas de estres usando varios clientes distribuidos	2008-12-27	1340	Bueno	6	
Análisis de rendimiento al usar un Pool de conexiones	Análisis de rendimiento de usar o no un pool de conexiones a bases de datos en nuestras aplicaciones	2008-12-25	1377	Muy bueno	15	
Utilización de grupos en Spring Security	En este primer tutorial que publico en Adictos al Trabajo os muestro cómo utilizar Spring Security para controlar el acceso a una aplicación utilizando grupos de usuarios.	2008-12-16	2080	Bueno	23	

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.