

¿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)



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

[Ver cursos que ofrece Autentia](#)

[Descargar comics en PDF y alta resolución](#)



[¡NUEVO!] 2008-11-17



2008-09-01

¿Por qué no se aplican los mismos criterios de la vida a la informática?



2008-07-31



2008-07-08

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la sección [comic](#).

Tutorial desarrollado por

Catálogo de servicios de Autentia



Jose Manuel Suárez Sánchez

Consultor tecnológico de desarrollo de proyectos informáticos. Diseñador de Adictos Al Trabajo 2.0

Puedes encontrarme en [Autentia](#)

Somos expertos en Java/J2EE

[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: SpringEmail.pdf](#)

Fecha de creación del tutorial: 2008-11-24

Integración de Spring con el envío de emails.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Maven: el pom.xml.
- 4. Creación de nuestro servicio de envío de emails.
- 5. Configuración del applicationContext.xml.
- 6. Creación de un test de JUnit.
- 7. Conclusiones.

1. Introducción

Siguiendo con la saga de tutoriales sobre Spring y el soporte a la integración con otras tecnologías que proporciona, vamos a analizar la integración con un servicio de correo electrónico.

Dentro del amplio catálogo de escenarios en el que podemos integrar Spring en nuestros desarrollos, solemos decir que no es una solución "todo o nada", de modo que nos podemos beneficiar de todo el conjunto de módulos que contiene para darnos soporte a la generación de una aplicación JEE o sólo de una parte de ellos.

En este tutorial vamos a implementar un servicio de envío de emails con attachments, al que tendremos acceso:

- desde cualquier módulo de nuestra aplicación gestionado por Spring, haciendo uso de su inyección de dependencias, o
- desde cualquier módulo de nuestra aplicación, que sin estar gestionado por Spring, pueda acceder al contenedor de IoC de Spring para recuperar el servicio.

Para comprobar el funcionamiento del servicio haremos un test de JUnit que levantará el contexto de Spring.

Catálogo de servicios Autentia (PDF 6,2MB)



[En formato comic...](#)



- Web
- www.adictosaltrabajo.com

Últimos tutoriales

2008-11-24 [Integración de Spring con el envío de emails](#)

2008-11-17 [Introducción a JTrac](#)

2008-11-17 [Cómo crear carruseles con detalle con jcarousel y jtip](#)

2008-11-08 [JPivot, como crear otro UI para especificar las dimensiones del cubo OLAP](#)

2008-11-05 [Cómo crear ventanas modales con modalbox](#)

2008-11-03 [Primeros pasos por el mundo Java/Eclipse \(en Vista\)](#)

2008-10-31 [Planificación de tareas en Java mediante Quartz](#)

2008-10-31 [GrassGames eBook Reader: leer PDF en pantalla cómodamente](#)

2008-10-30 [Autenticación y Autorización mediante JAAS](#)

2008-10-27

La redacción de este tutorial se realiza dando por hecho que el lector tiene conocimientos suficientes sobre [Spring IoC](#), [Maven](#) y [JUnit 4.4](#).

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Sobremesa Dell Dimension 6400, 2.13 Ghz, 2 Gb RAM
- Sistema operativo: Windows XP Media center Edition
- JDK 1.5.0_15
- Eclipse 3.4.

3. Maven: el pom.xml.

El soporte que proporciona Spring para el envío de emails no es más que un wrapper sobre las librerías javax.mail y javax.activation, con lo que, además de las dependencias típicas para Spring y el entorno de test incluimos la dependencia a estas dos librerías.

```
view plain print ?
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
02.     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0
03.     <modelVersion>4.0.0</modelVersion>
04.     <groupId>com.autentia.training.spring</groupId>
05.     <artifactId>SpringEmail</artifactId>
06.     <packaging>jar</packaging>
07.     <version>1.0-SNAPSHOT</version>
08.     <name>SpringEmail</name>
09.     <description>Ejemplo de uso de un servicio de Email en una aplicación con Spring.</descript
10.     <url>http://maven.apache.org</url>
11.     <dependencies>
12.         <!-- ===== Spring ===== -->
13.         <dependency>
14.             <groupId>org.springframework</groupId>
15.             <artifactId>spring</artifactId>
16.             <version>2.5.5</version>
17.         </dependency>
18.         <!-- para poder hacer uso de la anotación @Resource con la jdk1.5 -->
19.         <dependency>
20.             <groupId>javax.annotation</groupId>
21.             <artifactId>jsr250-api</artifactId>
22.             <version>1.0</version>
23.         </dependency>
24.         <!-- ===== Mail ===== -->
25.         <dependency>
26.             <groupId>javax.mail</groupId>
27.             <artifactId>mail</artifactId>
28.             <version>1.4</version>
29.         </dependency>
30.         <dependency>
31.             <groupId>javax.activation</groupId>
32.             <artifactId>activation</artifactId>
33.             <version>1.1</version>
34.         </dependency>
35.         <!-- ===== Test ===== -->
36.         <dependency>
37.             <groupId>junit</groupId>
38.             <artifactId>junit</artifactId>
39.             <version>4.4</version>
40.             <scope>test</scope>
41.         </dependency>
42.         <dependency>
43.             <groupId>org.springframework</groupId>
44.             <artifactId>spring-test</artifactId>
45.             <version>2.5.5</version>
46.             <scope>test</scope>
47.         </dependency>
48.     </dependencies>
49.     <build>
50.         <finalName>SpringEmail</finalName>
51.         <plugins>
52.             <plugin>
53.                 <artifactId>maven-compiler-plugin</artifactId>
54.                 <configuration>
55.                     <source>1.6</source>
56.                     <target>1.6</target>
57.                     <encoding>UTF-8</encoding>
58.                 </configuration>
59.             </plugin>
60.         </plugins>
61.     </build>
62. </project>
```

4. Creación de nuestro servicio de envío de emails.

Comenzamos con la implementación del servicio, creando una interfaz, para obligarnos a implementar una serie de métodos, los típicos para el envío de correo electrónico.

Últimas ofertas de empleo

2008-10-30
Comercial - Ventas - BARCELONA.

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

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

2008-10-03
Marketing - Experto en Marketing - MADRID.

2008-10-01
Atención a cliente - Call Center - MADRID.

Anuncios Google

```
view plain print ?
01. package com.autentia.training.spring.mail;
02.
03. import java.io.File;
04.
05. public interface MailService {
06.     public void send(String to, String subject, String text);
07.     public void send(String to, String subject, String text, File... attachments);
08.
09. }
10.
11. }
```

Ahora creamos una clase que implemente la interaz creada

view plain print ?

```
01. package com.autentia.training.spring.mail;
02.
03. import java.io.File;
04.
05. import javax.mail.MessagingException;
06. import javax.mail.internet.MimeMessage;
07.
08. import org.apache.commons.logging.Log;
09. import org.apache.commons.logging.LogFactory;
10. import org.springframework.core.io.FileSystemResource;
11. import org.springframework.mail.javamail.JavaMailSenderImpl;
12. import org.springframework.mail.javamail.MimeMessageHelper;
13. import org.springframework.util.Assert;
14.
15. /**
16.  * Servicio de envío de emails
17.  */
18. public class MailServiceImpl implements MailService {
19.
20.     private static final Log log = LogFactory.getLog(MailServiceImpl.class);
21.
22.     /** wrapper de Spring sobre javax.mail */
23.     private JavaMailSenderImpl mailSender;
24.
25.     public void setMailSender(JavaMailSenderImpl mailSender) {
26.         this.mailSender = mailSender;
27.     }
28.
29.     /** correo electrónico del remitente */
30.     private String from;
31.
32.     public void setFrom(String from) {
33.         this.from = from;
34.     }
35.
36.     public String getFrom() {
37.         return from;
38.     }
39.
40.     /** flag para indicar si está activo el servicio */
41.     public boolean active = true;
42.
43.     public boolean isActive() {
44.         return active;
45.     }
46.
47.     public void setActive(boolean active) {
48.         this.active = active;
49.     }
50.
51.     private static final File[] NO_ATTACHMENTS = null;
52.
53.     /** envío de email
54.      * @param to correo electrónico del destinatario
55.      * @param subject asunto del mensaje
56.      * @param text cuerpo del mensaje
57.      */
58.     public void send(String to, String subject, String text) {
59.         send(to, subject, text, NO_ATTACHMENTS);
60.     }
61.
62.     /** envío de email con attachments
63.      * @param to correo electrónico del destinatario
64.      * @param subject asunto del mensaje
65.      * @param text cuerpo del mensaje
66.      * @param attachments ficheros que se anexarán al mensaje
67.      */
68.     public void send(String to, String subject, String text, File... attachments) {
69.         // chequeo de parámetros
70.         Assert.hasLength(to, "email 'to' needed");
71.         Assert.hasLength(subject, "email 'subject' needed");
72.         Assert.hasLength(text, "email 'text' needed");
73.
74.         // asegurando la trazabilidad
75.         if (log.isDebugEnabled()) {
76.             final boolean usingPassword = !" ".equals(mailSender.getPassword());
77.             log.debug("Sending email to: '" + to + "' [through host: '" + mailSender.getHost()
78.                 + mailSender.getPort() + "', username: '" + mailSender.getUsername() + "' \
79.                 + usingPassword + "].");
80.             log.debug("isActive: " + active);
81.         }
82.         // el servicio esta activo?
83.         if (!active) return;
84.
85.         // plantilla para el envío de email
86.         final MimeMessage message = mailSender.createMimeMessage();
87.
88.         try {
```

El código está comentado, pero añadimos lo siguiente:

- **líneas 23 a 27:** la inclusión del servicio de envío propio de Spring y la espera de ser inyectado mediante el método set,
- **líneas 29 a 38:** la asignación de un remitente de correo por defecto para todos nuestros emails, será también inyectado por Spring,
- **línea 86:** creación de la plantilla para el envío de emails. La clase MimeMessage tiene lo mínimo para el envío de emails a través del servicio de Spring.
- **línea 90:** creación de la plantilla para el envío de emails con attachments, permite multipart.
- **líneas 92 a 107:** poblamos las plantilla con los parámetros del método y el remitente por defecto,
- **línea 114:** el envío propiamente dicho a través del servicio de Spring.

5. Configuración del applicationContext.xml.

Publicamos nuestro servicio en el contexto del **applicationContext.xml**, inyectándole la dependencia al servicio de envío propio de Spring y el remitente por defecto. Para la configuración del servicio de Spring nos apoyamos en un fichero de configuración administrable: **app.properties**, que tendrá los datos de conexión con el servidor de correo.

```
view plain print ?
01. <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XI
02.     xmlns:util="http://www.springframework.org/schema/util" xmlns:context="http://www.springfr
03.     xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework
04.     xsi:schemaLocation="
05.         http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
06.         http://www.springframework.org/schema/util http://www.springframework.org/schema/util/s
07.         http://www.springframework.org/schema/context http://www.springframework.org/schema/cor
08.         http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spi
09.         http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spri
10.
11.     <context:annotation-config />
12.
13.     <context:component-scan base-package="com.autentia.training" />
14.
15.     <!--#lectura del fichero administrable -->
16.     <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlace
17.         <property name="ignoreUnresolvablePlaceholders">
18.             <value>true</value>
19.         </property>
20.         <property name="locations">
21.             <list>
22.                 <value>classpath:app.properties</value>
23.             </list>
24.         </property>
25.     </bean>
26.
27.     <!--#Configuración del servicio de Spring: MailSernder -->
28.     <bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
29.         <property name="host" value="${mail.host}" />
30.         <property name="port" value="${mail.port}" />
31.         <property name="username" value="${mail.username}" />
32.         <property name="password" value="${mail.password}" />
33.         <property name="defaultEncoding" value="UTF-8" />
34.     </bean>
35.
36.     <!--#Configuración de nuestro servicio:MailService -->
37.     <bean id="mailService" class="com.autentia.training.spring.mail.MailServiceImpl">
38.         <property name="active" value="true" />
39.         <property name="mailSender" ref="mailSender" />
40.         <property name="from" value="default@unknown.com" />
41.     </bean>
42.
43. </beans>
```

Destacamos lo siguiente:

- **líneas 12 y 14:** configuración del contexto de Spring para el uso de anotaciones,
- **líneas 16 a 26:** configuración de un servicio de tipo PropertyPlaceholderConfigurer para la lectura de ficheros de propiedades externos, de modo que, como si accediesemos a variables, obtener el valor de las claves declaradas en el fichero de properties,
- **líneas 29 a 35:** configuración del servicio de Spring para el envío, al que inyectamos la configuración sobre el servidor de correo, obtenida del fichero de propiedades. Para comprobar qué podemos configurar del servicio basta con revisar el [javadoc de la clase](#) de modo que cada método set de la clase coincide con uno de nuestros <property (inyección de dependencias a través de propiedades),
- **líneas 38 a 42:** configuración de nuestro servicio de envío inyectándole el servicio de Spring y el remitente por defecto.

El código de nuestro fichero de propiedades **app.properties**:

```
view plain print ?
01. mail.host=smtpt.unknown.com
02. mail.port=25
03. mail.username=user
04. mail.password=passw
```

Con la implementación de la interfaz y esta configuración ya podemos hacer uso de nuestro servicio de envío desde cualquiera de nuestras clases de negocio gestionadas por Spring mediante la anotación @Resource.

6. Creación de un test de JUnit.

Vamos a probar su funcionamiento a mediante un test de JUnit. El test comprobará un error en el envío, puesto que la configuración de nuestro host es incorrecta, pero nos sirve para ilustrar su funcionamiento y cómo realizar la inyección.

```
view plain print ?
01. package com.autentia.training.spring.mail;
02.
03. import javax.annotation.Resource;
04.
05. import junit.framework.Assert;
06.
07. import org.apache.commons.logging.Log;
08. import org.apache.commons.logging.LogFactory;
09. import org.junit.Test;
10. import org.junit.runner.RunWith;
11. import org.springframework.test.context.ContextConfiguration;
12. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
13.
14. @RunWith(SpringJUnit4ClassRunner.class)
15. @ContextConfiguration(locations = { "/applicationContext.xml" })
16. public class MailServiceTest {
17.
18.     private static final Log log = LogFactory.getLog(MailServiceTest.class);
19.
20.     @Resource
21.     private MailService mailService;
22.
23.     /**
24.      * Probamos el envío
25.      */
26.     @Test
27.     public void cantSendMails() {
28.         try {
29.             mailService.send("jmsanchez@autentia.com", "Test de envío de email.", "Prueba del e
30.             Assert.fail("No debería realizar el envío puesto que el host no está correctamente
31.         }
32.         catch(Exception e){
33.             log.trace("Excepción controlada, normal en el entorno de test",e);
34.         }
35.     }
36. }
37. }
```

Para la comprobación del correcto funcionamiento tendríamos que retocar el código de nuestro test.

Destacamos las anotaciones de la clase (líneas 14 y 15) que nos permiten levantar desde la ejecución del test el contexto de Spring, leyendo la configuración del fichero applicationContext.xml.

7. Conclusiones.

Sencilla la configuración de un servicio de envío de emails con Spring, ¿verdad?. Si además podemos hacer uso de la inyección de dependencias tendremos el servicio disponible desde cualquier clase de negocio.

El hecho de tener la configuración del servicio desacoplada mediante Spring, nos permitirá llevar a cabo una configuración del servidor smtp en función del entorno y, sobre todo, en función del entorno de tests.

Un saludo.

Jose

<mailto:jmsanchez@autentia.com>

- Puedes opinar sobre este tutorial [haciendo clic aquí](#).
- Puedes firmar en nuestro libro de visitas [haciendo clic aquí](#).
- Puedes asociarte al grupo AdictosAITrabajo 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

J2EE, EJBs, Struts...

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subscripción a novedades:

E-mail

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	pdf
SpringIDE, plugin de Spring para Eclipse	En adictosaltrabajo os hemos ido presentando diversos plugins para Eclipse. Esta vez le toca el turno a SpringIDE, un plugin que os ayudará a desarrollar aplicaciones que utilicen Spring.	2008-01-19	4870	pdf
Crear un logger utilizado a través de aspectos con Spring AOP.	En este tutorial os enseñamos cómo implementar un logger utilizado a través de aspectos con Spring AOP.	2008-02-22	2191	pdf
Creación de una aplicación web con SpringMVC desde 0	Este tutorial te resultará muy útil para aprender a usar el patrón modelo-vista-controlador (MVC) con Spring a nuestros desarrollos web	2008-05-05	3966	pdf
Spring WebFlow con Validator	En este tutorial se muestra como podemos realizar las validaciones más frecuentes de datos mediante Spring WebFlow.	2007-12-11	3437	pdf
Creación de una aplicación con Spring e Hibernate desde 0	Este tutorial vamos a explicar paso a paso cómo crear una pequeña aplicación usando Spring e Hibernate con anotaciones partiendo desde 0	2008-02-15	8162	pdf
Cómo realizar pruebas unitarias con Spring y JUnit4 utilizando Gienah	En este tutorial vamos a presentaros Gienah, una tecnología que os permitirá de una forma muy cómoda y sencilla utilizar componentes de Spring en vuestros test unitarios realizados con JUnit 4	2008-02-17	1709	pdf
Spring + Hibernate + Anotaciones = Desarrollo Rápido en Java	Alejandro Pérez nos enseña lo fácil y rápido que es desarrollar en Java usando Spring e Hibernate, y usando anotaciones	2008-05-14	7606	pdf
Spring WebFlow Tiles	En este tutorial aprenderemos el uso de tiles para usarlo en conjunción con Spring WebFlow.	2007-11-26	3049	pdf
Manual básico de Spring WebFlow	En este tutorial Javier Antoniucci nos enseña cómo empezar a trabajar con el framework de desarrollo web Spring webflow.	2007-11-26	4237	pdf
Planificación de tareas con Spring	Spring nos proporciona varias formas de planificar las tareas a través de Quartz, y en este tutorial Juan nos enseña un ejemplo práctico	2008-10-10	1003	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.