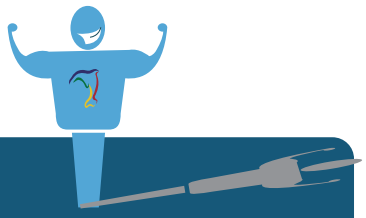
autentia

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...

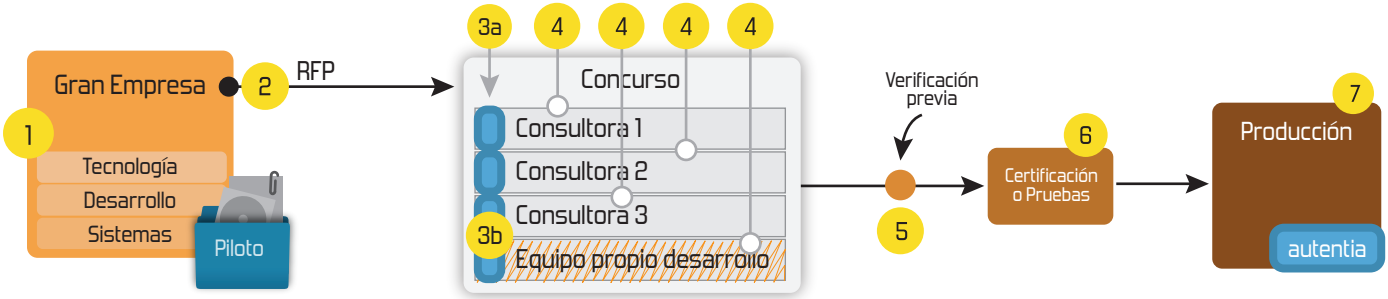
1. Desarrollo de componentes y proyectos a medida



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

Tutorial desarrollado por

Jose Manuel Sánchez Suárez

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

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: SpringEmailInlineResources.pdf

Fecha de creación del tutorial: 2008-12-05

Integración de Spring con el envío de emails: técnicas avanzadas (I).

0. Índice de contenidos.

1. Introducción

Este tutorial es un complemento al publicado recientemente sobre el envío de email mediante el soporte que proporciona Spring, de hecho se basa en el ejemplo que se plantea de creación de un servicio de envío de correo electrónico.

Spring nos ofrece un emisor de correo electrónico y una familia de plantillas, que cubren todas nuestras necesidades de envío. En este tutorial vamos a realizar una prueba de concepto de la plantilla que nos permite incrustar recursos dentro del fuente de nuestros correos electrónicos. Ya hemos visto como podemos adjuntar un documento, ahora vamos a enviar un email con formato html, incrustando ciertos recursos en línea para la correcta visualización del mensaje.

La implementación del emisor en Spring no es más que un wrapper de javax.mail, si no disponéis de Spring en vuestros proyectos siempre podéis echar un vistazo a este tutorial sobre el envío de correo electrónico mediante javax.mail.

Últimos tutoriales

Para comprobar el funcionamiento del servicio añadiremos un método al test de JUnit que levantará el contexto de Spring.

La redacción de este tutorial se realiza dando por hecho que el lector tiene conocimientos suficientes sobre Spring IoC, Maven y JUnit 4.4 y ha realizado la lectura del tutorial en el se basa este.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Asus G1 (Core 2 Duo a 2.1 GHz, 2048 MB RAM, 120 GB HD).
- Sistema operativo: Windows Vista Ultimate.
- JDK 1.5.0_15
- Eclipse 3.4, con q4e.

3. Maven: el pom.xml.

El pom.xml es el planteado para el ejemplo del primer tutorial, añadiendo una librería adicional que no es estríctamente necesaria.

```
view plain  print  ?
01. <project ... >
02. ...
03.         <dependency>
04.             <groupId>org.springframework.batch</groupId>
05.             <artifactId>spring-batch-infrastructure</artifactId>
06.             <version>1.1.3.RELEASE-A</version>
07.         </dependency>
08. ...
09. </project>
```

Spring batch es un proyecto de Spring framework que tiene como objetivo principal proporcionar los servicios e interfaces comúnes que cualquier aplicación batch puede utilizar.

Nosotros hacemos uso de una clase de utilidades para la lectura de ficheros planos, si bien, podéis hacer uso de las técnicas que estéis utilizando en la actualidad para realizar dichas lecturas, y prescindir de esta librería.

4. Modificación del servicio de envío de emails.

Vamos a modificar la interfaz MailService que establece cómo deben ser las clases de servicio para el envío de emails en nuestra aplicación añadiendo un nuevo método.

```
view plain  print  ?
01. ...
02.   public void send(String to, EmailMessageProvider messageProvider);
03. ...
```

La idea es tener una familia de proveedores de mensajes que, en función de la implementación concreta, implementen la lógica de obtención de los recursos para el envío del email de donde sea necesario. De esta forma desacoplamos la construcción del mensaje del propio envío.

La nueva definición del método en la interfaz nos obliga a implementarlo en nuestra clase de servicio MailServiceImpl:

```
view plain  print  ?
...

    /** envío de email
      * @param to correo electrónico del destinatario
      * @param messageProvider messageProvider
      */
    public void send(String to, EmailMessageProvider messageProvider) {
        // chequeo de parámetros
        Assert.hasLength(to, "email 'to' needed");
        Assert.notNull(messageProvider);

        // asegurando la trazabilidad
        if (log.isDebugEnabled()) {
            final boolean usingPassword = !"".equals(mailSender.getPassword());
            log.debug("Sending email to: '" + to + "' [through host: '" + mailSender.getHost()
                    + mailSender.getPort() + "', username: '" + mailSender.getUsername() + "' u
                    + usingPassword + "].");
            log.debug("isActive: " + active);
        }
        // el servicio esta activo?
        if (!active) return;

        // plantilla para el envío de email
        final MimeMessage message = mailSender.createMimeMessage();

        try {
            // el flag a true indica que va a ser multipart
            final MimeMessageHelper helper = new MimeMessageHelper(message,true);
            helper.setTo(to);
            helper.setSubject(messageProvider.getSubject());
            helper.setFrom(getFrom());
            // el flag a true indica que el cuerpo del mensaje es HTML
            helper.setText(messageProvider.getBody(),true);

            // añadiendo los ficheros "en línea"
            if (messageProvider.getInlineFiles() != null) {
                for (String key : messageProvider.getInlineFiles().keySet()) {
                    Resource value = messageProvider.getInlineFiles().get(key);
                    helper.addInline(key, value);
                    if (log.isDebugEnabled()) {
                        log.debug("File '" + value + "' added.");
                    }
                }
            }

        } catch (MessagingException e) {
            new RuntimeException(e);
        }

        // el envío
        this.mailSender.send(message);
    }
...
```

El código es similar al método creado, en el primer tutorial, en la misma clase, que envia archivos adjuntos. Está comentado, pero añadimos lo siguiente:

- **línea 33**: el método setText acpeta un segundo parámetro que indica si el cuerpo del mensaje es HTML,
- **líneas 35 a 44**: recorremos el listado de recursos en línea del proveedor de mensajes para añadirlos con la clave establecida al cuerpo del mensaje,

5. Creación del proveedor de mensajes.

Vamos a ver qué contiene la definición de la interfaz del proveedor de mensajes:

```
view plain  print  ?
package com.autentia.training.spring.mail;

import java.util.Map;

import org.springframework.core.io.Resource;

public interface EmailMessageProvider {

    public String getSubject();

    public String getBody();

    public Map<String, Resource> getInlineFiles();

}
```

El proveedor de mensajes encapsula el asunto, el cuerpo y un mapa con las claves y los contenidos a añadir al cuerpo del mensaje.

Ahora veremos una clase de implementación de dicha interfaz:

view plain print ?

```java
package com.autentia.training.spring.mail;

import java.io.IOException;
import java.util.Map;

import org.springframework.batch.item.file.separator.ResourceLineReader;
import org.springframework.core.io.Resource;

public class SimpleEmailMessageProviderImpl implements EmailMessageProvider {

    private Map<String,Resource> inlineFiles;

    private String subject;

    private String body;

    @SuppressWarnings("unused")
    private Resource template;

    public String getSubject(){
        return subject;
    }

    public void setSubject(String subject){
        this.subject = subject;
    }

    public String getBody() {
        return body;
    }

    public Map<String, Resource> getInlineFiles() {
        return inlineFiles;
    }

    public void setInlineFiles(Map<String, Resource> inlineFiles) {
        this.inlineFiles = inlineFiles;
    }

    public void setTemplate(Resource template) throws IOException {
        StringBuffer fileContent = new StringBuffer();
        ResourceLineReader reader = new ResourceLineReader(template);
        String line;
        while ((line = (String) reader.read()) != null) {
            fileContent.append(line);
        }
        this.body = fileContent.toString();
    }
}
```

No es más que un POJO con un método extra: setTemplate que recibe como parámetro un Resource (la interfaz Resource de Spring Framework no es más que un wrapper de File o de URL, según el caso concreto). Se apoya en la clase de utilidades ResourceLineReader de Spring Batch para realiar la lectura del contenido del recurso y asignárselo al cuerpo del mensaje.

¿De donde obtenemos el contenido del mensaje?, nos vendrá dado, inyectado, por la configuración del contenedor de Spring.

6. Configuración del applicationContext.xml.

Vamos a añadir la definición de nuestro proveedor de mensajes dentro del **applicationContext.xml**:

view plain print ?

```xml
<beans  ... >
...
    <!--#Configuración de nuestro proveedor de mensaje: EmailMessageProvider -->
    <bean id="emailMessageProviderImpl" class="com.autentia.training.spring.mail.SimpleEmailMes
      <property name="subject" value="${email.subject.trainingConfirmation}"/>
      <property name="template" value="classpath:email/trainingConfirmation.html"/>
      <property name="inlineFiles">
        <map>
            <entry>
                <key>
                    <value>id001</value>
                </key>
                <value>classpath:generic.css</value>
            </entry>
            <entry>
                <key>
                    <value>id002</value>
                </key>
                <value>classpath:autentia.gif</value>
            </entry>
        </map>
      </property>
    </bean>
...
</beans>
```

Destacamos lo siguiente:

- **línea 6**: el asunto lo obtenemos añadíendolo al fichero de propiedades que teníamos definido: app.properties
- **línea 7**: la plantilla la obtenemos de un fichero html creado al efecto dentro de la carpeta de recursos
- **líneas 8 a 23**:los recursos a incrustar en el cuerpo del mensaje los pasamos dentro de un mapa de clave:valor, donde la clave es un identificador único y el valor la obtención de un recurso de classpath. Vamos a probar a añadir dos recursos: una css y una imagen.

El código de nuestro fichero de propiedades **app.properties** ahora tiene ademas esta clave:

```
view plain  print  ?
01. email.subject.trainingConfirmation=Confirmación de celebración de la charla sobre 'Spring Frame
```

7. La plantilla HTML.

La carpeta de recursos de nuestro proyecto tendrá la siguiente estructura de directorios:

La plantilla trainingConfirmation.html, puede tener un contenido parecido al que sigue:

```
view plain  print  ?
01. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTI
02. <html xmlns="http://www.w3.org/1999/xhtml">
03. <head>
04. <style type="text/css" media="screen">
05.  @import url(cid:id001);
06. </style>
07. <title>Confirmación de la celebración de la charla sobre "Spring Framework"</title>
08. </head>
09. <body>
10.     <div id="header">
11.         <img src="cid:id002" />
12.     </div>
13.     <div id="content">
14.         <p>Hola,</p>
15.         <p>Te confirmamos la celebración de la charla sobre "Spring Framework" del 22 de enero
16.         <p>Un saludo.</p>
17.         <p>El equipo de Autentia.</p>
18.     </div>
19.     <div id="footer">
20.         <p>De conformidad con lo establecido en la legislación vigente en materia de Protecció
21.         (Ley Orgánica de Protección de Datos, 15/1999 de 13 de Diciembre), la empresa Autentia
22.         comunica al receptor de éste correo electrónico que sus datos personales forman parte
23.         titularidad de la empresa, cuya finalidad es el mantenimiento de la relación empresari
24.         su empresa: envío de información y publicidad sobre ofertas de servicio, promociones,
25.         de cursos y seminarios.</p>
26.
27.         <p>Usted podrá ejercitar los derechos de acceso, rectificación, oposición y cancelació
28.         o correo electrónico a Autentia Real Business Solutions S.L. info@autentia.com. Avda.
29.         Parque Empresarial San Fernando 28830 San Fernando de Henares. Madrid.</p>
30.     </div>
31. </body>
32. </html>
```

Las claves únicas que habíamos declarado en el mapa de recursos, en la definición del proveedor de mensajes, dentro del applicationContext.xml, las tenemos aquí asignadas:

- **en la línea 5**: en la importación de la hoja de estilos css,
- **en la línea 11**: en el fuente de la imagen a mostrar en la cabecera del mensaje.

La imagen se corresponde con el logo de la empresa y la css puede tener un contenido parecido al siguiente:

```
view plain  print  ?
01. body {
02.     font-family:Verdana;
03. }
04. div {
05.     margin:20px;
06. }
07. #header {
08.     font-size:1.2em;
09.     padding-bottom:10px;
10.     border-bottom:1px solid #123456;
11. }
12. #content {
13.     font-size:0.9em;
14. }
15. #footer {
16.     font-size:0.7em;
17.     padding-top:10px;
18.     border-top:1px solid #123456;
19. }
```

8. Creación de un nuevo testCase.

Vamos a probar su funcionamiento añadiendo al test:

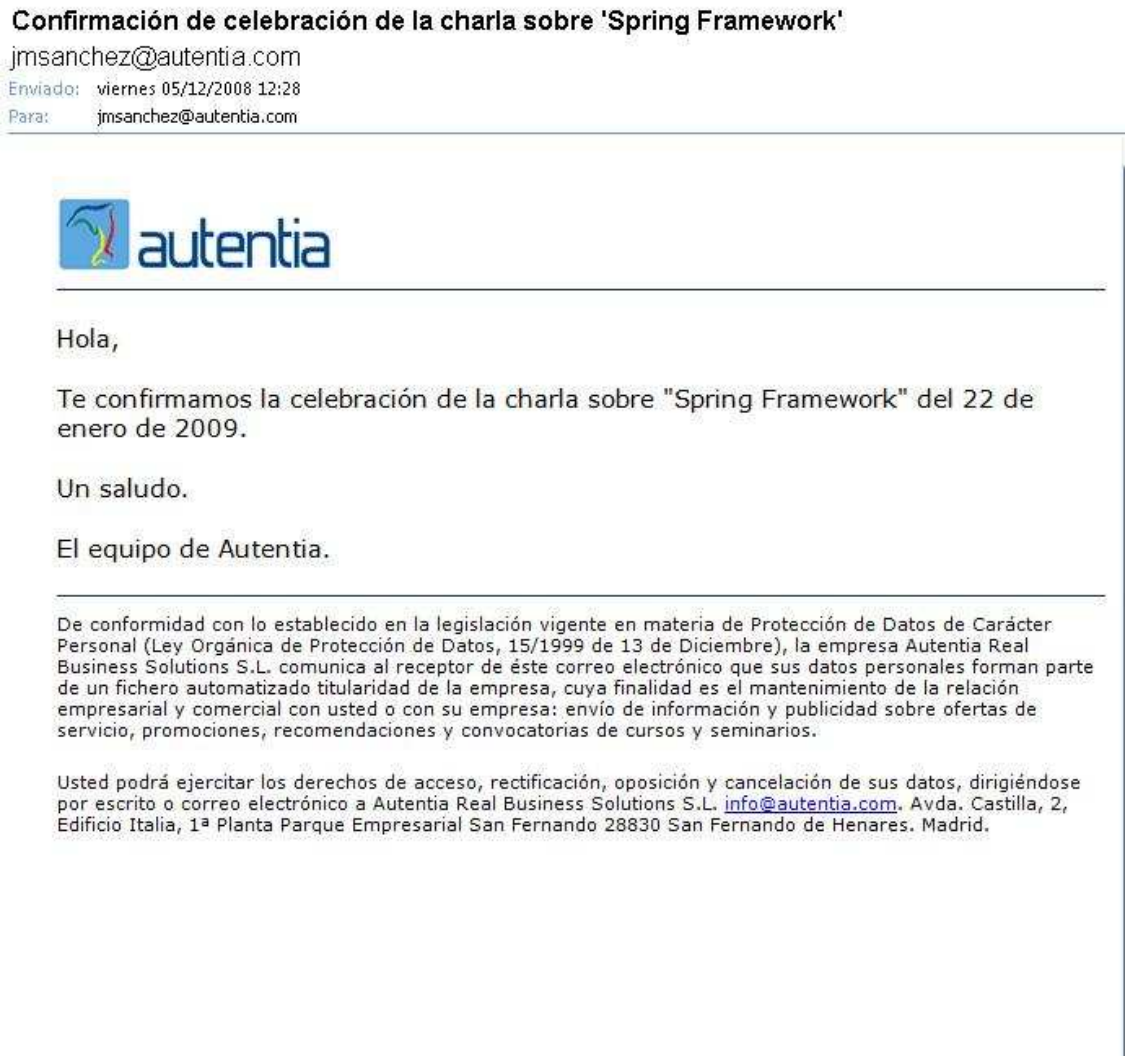
- la inyección del proveedor de mensajes (**línea 4**), y
- un nuevo método que realice la invocación (**líneas 11 a 20**).

```
view plain  print  ?
01. ...
02.
03.   @Resource
04.     private EmailMessageProvider emailMessageProvider;
05.
06.     /**
07.       * Probamos el envío con imagenes incrustadas
08.       * @throws MessagingException
09.       */
10.     @Test
11.     public void testInlineResources() throws MessagingException {
12.
13.         try {
14.             mailService.send("jmsanchez@autentia.com", emailMessageProvider);
15.         }
16.         catch(Exception e){
17.             final String msg = "Excepción en el envío de emails con recursos incrustados.";
18.             log.warn(msg,e);
19.             Assert.fail(msg);
20.         }
21.
22.     }
23. ...
```

El hecho de tener desacoplado tanto el proveedor de mensajes como el servicio de envío nos permitiría simular su funcionamiento, desde el entorno de test, creándo nuestros propios Mock Objects.

10. El resultado.

Tras la ejecución del test deberíamos tener en la bandeja de entrada de nuestro cliente de correo un mensaje como el que sigue:



Confirmación de celebración de la charla sobre 'Spring Framework'

jmsanchez@autentia.com

Enviado: viernes 05/12/2008 12:28

Para: jmsanchez@autentia.com

autentia

Hola,

Te confirmamos la celebración de la charla sobre "Spring Framework" del 22 de enero de 2009.

Un saludo.

El equipo de Autentia.

De conformidad con lo establecido en la legislación vigente en materia de Protección de Datos de Carácter Personal (Ley Orgánica de Protección de Datos, 15/1999 de 13 de Diciembre), la empresa Autentia Real Business Solutions S.L. comunica al receptor de éste correo electrónico que sus datos personales forman parte de un fichero automatizado titularidad de la empresa, cuya finalidad es el mantenimiento de la relación empresarial y comercial con usted o con su empresa: envío de información y publicidad sobre ofertas de servicio, promociones, recomendaciones y convocatorias de cursos y seminarios.

Usted podrá ejercitar los derechos de acceso, rectificación, oposición y cancelación de sus datos, dirigiéndose por escrito o correo electrónico a Autentia Real Business Solutions S.L. info@autentia.com. Avda. Castilla, 2, Edificio Italia, 1ª Planta Parque Empresarial San Fernando 28830 San Fernando de Henares. Madrid.

Tanto la visualización en formato HTML como la de los estilos dependerá del cliente de correo que estemos usando, con gmail no parece que se puedan incrustar css, de ahí que quizás la maquetación del cuerpo del mensaje en HTML no podamos dejarla en manos de una hoja de estilos externa. Para probar la funcionalidad que nos ocupa nos basta.

9. Conclusiones.

Ampliamos la funcionalidad básica que planteábamos en el primer tutorial y comprobamos que, si tenemos todo bien desacoplado, nos resultará sencillo extender.

Tratamos de sentar las bases para cumplir con el principio: "Open for extension, closed for modifications", de modo que nuestro código sea cada vez más mantenible al estar menos acoplado, por ende más reusable, más fácilmente configurable y más escalable.

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 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

Gestión de contenidos

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	pdf
HtmlEmail. Envío de emails en HTML con imágenes embebidas	Ejemplo del uso del API commons email para enviar correos HTML con imagenes embebidas	2008-02-13	2495	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	2264	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	8375	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	1805	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	4146	pdf
Integración de Spring con el envío de emails	Nuestro compañero Jose, continuando con la saga de tutoriales de Spring, nos enseña en ésta ocasión la integración con un servicio de correo electrónico	2008-11-24	490	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	1165	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	7956	pdf
JavaMail: Envía e-mails desde tu proyecto Java	En éste primer tutorial que publico en ésta prestigiosa Web os enseño a como podeis integrar las librerías JavaMail para el envío de correos electrónicos en tu proyecto Java	2008-12-01	310	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	5082	pdf

Nota: