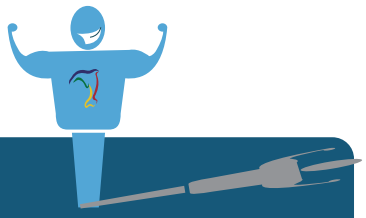
autentia

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...

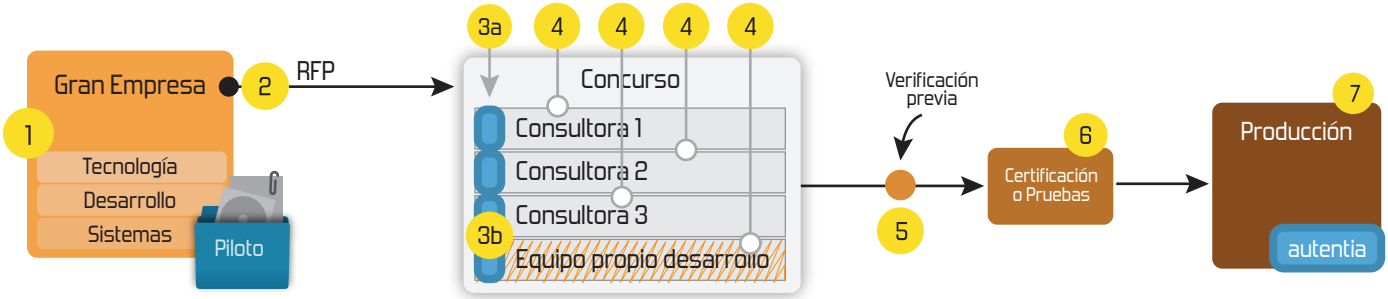
1. Desarrollo de componentes y proyectos a medida



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

Ultimas Noticias

» Estuvimos en el evento de Liferay en Madrid
» VII Charla Autentia - Pluto
» Competición Plasma Cars (Autos Locos) - SEGUNDO INTENTO
» Probando con Marick - Fotos y vídeo
» Competición Plasma Cars (Autos Locos) - EVENTO POSPUESTO
» VI Charla Autentia: Mapeos en Hibernate - Vídeos y Material
» Competición Plasma Cars (Autos Locos) - EVENTO

+Noticias Destacadas

» Estuvimos en el evento de Liferay en Madrid
» Competición Plasma Cars (Autos Locos) - SEGUNDO INTENTO
» Probando con Marick - Fotos y vídeo
» VI Charla Autentia: Mapeos en Hibernate - Vídeos y Material

+Comentarios Cómic

+Enlaces

Tutorial desarrollado por

Juan Alonso Ramos

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática de Gestión (cursando Ingeniería Informática)

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Descargar este documento en formato PDF: JCaptcha.pdf

Fecha de creación del tutorial: 2010-04-26

JCaptcha - Generación de Captchas en Java

Índice de contenidos.

1. Introducción

En este tutorial veremos la librería JCaptcha. Se trata de una librería opensource para generación de captchas en Java. En todo formulario web que pida al usuario introducir datos que posteriormente serán procesados deberíamos introducir una imagen de tipo Captcha que nos asegure que quién está enviando el formulario es un humano y no una máquina. Si no añadimos esta característica no podríamos evitar que un robot o programa automatizado pudiera darse de alta repetidas veces en nuestra web, participar en encuestas, etc.

JCaptcha nos aporta las herramientas necesarias para generar las imágenes que utilizaremos para validar nuestros formularios proporcionándonos además la personalización de la imagen permitiéndonos cambiar su tamaño, apariencia, fuente, colores, etc.

Otra de las grandes ventajas de esta librería es que se encuentra en los repositorios públicos de Maven por lo que con añadir la dependencia en nuestro pom.xml bastaría y por otro lado se integra con Spring por lo que su configuración es muy sencilla.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Asus G50Vseries (Core Duo P8600 2.4GHz, 4GB RAM, 320 GB HD).
- Sistema operativo: Windows 7
- ICEFaces 1.8.2
- JCaptcha 1.0
- Maven 2.2.1
- Spring 2.5.6
- Framework wuija 1.9

2. pom.xml

Para proyectos con Maven basta con añadir la dependencia en el pom.xml. Suponemos que en el pom.xml ya tenemos configuradas las dependencias a Spring, JSF, ICEFaces, etc.

Últimos tutoriales

2010-04-26
JCaptcha - Generación de Captchas en Java

2010-04-23
Instalar Puente PHP-Java en Tomcat

2010-04-22
Android: Ejemplo usando Widget, BroadcastReceiver y Localización

2010-04-20
Facelets en JSF 2: sistema de plantillas y componentes por composición.

```
view plain print ?
01.  < dependency>
02.     < groupId>com.octo.captcha</groupId>
03.     < artifactId>jcaptcha</artifactId>
04.     < version>1.0</version>
05.  < /dependency>
```

4. Configuración básica de JCaptcha con Spring

Existen varias formas de configurar JCaptcha en un proyecto web. Por un lado podemos desplegar un Servlet (SimpleImageCaptchaServlet) que atiende las peticiones del captcha, ver documentación aquí. Por otro lado podemos configurar los servicios necesarios para generación y validación del captcha a través de Spring. En este tutorial seguiremos esta configuración.

En el fichero de configuración de Spring applicationContext.xml añadimos el servicio de generación de las imágenes:

```
view plain print ?
01.  <bean class="com.octo.captcha.service.image.DefaultManageableImageCaptchaService id="imageCaptchaService"
```

Con esto ya tenemos disponible este servicio para la generación de la imagen del captcha, pero eso sí con su configuración por defecto que en muchos casos nos puede ser más que suficiente. En nuestra aplicación debemos crear la lógica de negocio, utilizando el servicio levantado anteriormente, para crear una imagen de captcha, validar la palabra 'oculta' del captcha, etc. Para ello crearemos un Managed Bean de JSF gestionado por String mediante la anotación @Controller y una clase Captcha que se encargará de todo lo relacionado con las operaciones del captcha.

La clase Captcha queda así:

2010-04-19
DbVisualizer free version.

2010-04-09
Session TimeOut en RichFaces, con el soporte de Jboss Seam.

2010-04-08
Jetspeed-2 de Apache Software Foundation

2010-04-07
Primeros pasos con Balsamiq Mockups

2010-03-18
Revisando los ejemplos de Cocos2d para IPhone.

2010-03-16
Organización de eventos con StageHQ

2010-03-15
Retrasar la carga de Javascript con jQuery.getScript().

2010-03-15
Optimización de páginas web con Page Speed.

2010-03-09
JSF 2 ya está aquí !!! The JSF Return, ahora más sencillo que nunca !!!

2010-03-08
Instalación de tus programas en tu IPhone.

2010-03-04
Sacar Release de un proyecto con Maven

2010-03-03
Instalación de Subversion y Apache en Ubuntu

2010-03-03
Cómo instalar la JDK de SUN en Fedora Linux

2010-03-02
Creando un botón de compra de Paypal con datos cifrados

2010-03-01
Creación de un plugin de tipo hook en Liferay

2010-03-01
ScrumCards de Autentia en Android

2010-02-25
Creando la baraja de SCRUM de Autentia como aplicación para Android

2010-02-25
Instalar CentOS en Virtualbox con NetInstall

view plain print ?

```java
package com.autentia.tutoriales.jsf;

import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.octo.captcha.service.image.DefaultManageableImageCaptchaService;
import com.octo.captcha.service.image.ImageCaptchaService;
import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;

public class Captcha {

    private final Log log = LogFactory.getLog(Captcha.class);

    private final DefaultManageableImageCaptchaService imageCaptchaService;

    private String captchaId;

    private String captchaValue;

    private byte[] captchaImage;

  /**
    * Recibe el servicio de generación de captcha y genera una nueva imagen
    */
    public Captcha(ImageCaptchaService imageCaptchaService) {
        this.imageCaptchaService = (DefaultManageableImageCaptchaService)imageCaptchaService;

        try {
            generateCaptchaImage();
        } catch (IOException e) {
            log.error("Error generando captcha: ' + e.getMessage());
        }
    }

    /**
      * Devuelve una imagen de captcha
      *
      * @return
      * @throws IOException
      */
    public void generateCaptchaImage() throws IOException {
        final HttpServletRequest httpServletRequest = (HttpServletRequest)FacesContext.getCurrentInstance
                .getExternalContext().getRequest();

        // Stream de salida para la imagen del captch
        final ByteArrayOutputStream jpegOutputStream = new ByteArrayOutputStream();

        // Guarda el identificador de sesión del usuario para validar el captch
        captchaId = httpServletRequest.getSession().getId();

        // Creación de la imagen de captchacall the ImageCaptchaService getChallenge metho
        final BufferedImage challenge = imageCaptchaService.getImageChallengeForID(captchaId);

        // Codificamos la imagen en JPE
        final JPEGImageEncoder jpegEncoder = JPEGCodec.createJPEGEncoder(jpegOutputStream);
        jpegEncoder.encode(challenge);

        // Guardamos la imagen como un flujo de bytes para pintarla en pantall
        captchaImage = jpegOutputStream.toByteArray();
    }

    /**
      * Validación del captcha. Se recoge el identificador de la sesión del usuario y el
      * reconocido el texto del captcha.
      *
      * @return
      */
    public boolean isValidCaptcha() {
        boolean validate = false;

        try {
            validate = imageCaptchaService.validateResponseForID(captchaId, captchaValue);
        } catch (Exception e) {
            log.error("Error durante la validación del captcha: ' + e.getMessage());
        }

        return validate;
    }

    public String getCaptchaValue() {
        return captchaValue;
    }

    public void setCaptchaValue(String captchaValue) {
        this.captchaValue = captchaValue;
    }

    public byte[] getCaptchaImage() {
        return captchaImage;
```


2010-02-22
Expresiones CRON

2010-02-19
Cómo utilizar el DataStore de Google App Engine con JDO

2010-02-19
Recursos Freeware

2010-02-17
Plugin de mejora de graficos para JMeter

2010-02-17
Cómo utilizar el datastore de Google App Engine con su API de nivel inferior

2010-02-16
Aprendiendo Objetive-C desarrollando para nuestro Iphone 3Gs

2010-02-11
Introducción a JCL.

2010-02-09
Creando la Baraja de SCRUM de Autentia como aplicación para el IPhone 3G.

2010-02-08
Cómo generar versiones imprimibles de páginas web

2010-02-04
Como cambiar el tamaño de las fuentes en Xcode (el entorno de desarrollo para Mac e iPhone)

2010-02-04
Primeros pasos con Enterprise Architect y UML 2.x

2010-02-04
Creación de un componente JSF, basádonos en un plugin de jQuery, con el soporte de RichFaces.

2009-02-03
Sincronizando el Mail de Mac con Gmail, el correo de Google

2010-02-03
Integración de jQuery en RichFaces.

2010-02-02
AjaxSingle: el partialSubmit de RichFaces.

2010-02-01
Introducción a RichFaces.

2010-01-29
Transformación de mensajes en SOA con OpenESB

2010-01-26
JMeter. Uso de funciones.

El controlador de la vista UserCtrl se encargará de recuperar los datos del formulario y validar el captcha.

2010-01-18
Autenticando los usuarios de Sonar contra un LDAP

2010-01-18
Introducción a jQuery UI.

2010-01-18
jQuery: cómo crear nuestros propios plugins.

2010-01-18
Cómo consumir un servicio web RESTful con el soporte de Ajax y JSON de jQuery.

2010-01-18
Introducción a jQuery.

2010-01-17
Introducción a Tapestry 5

2010-01-14
JMeter. Gestión de usuarios

2010-01-14
Patrón Visitor con commons-collections y sus Closures

2010-01-12
Creación de servicios web RestFul, con soporte a persistencia, en NetBeans.

2010-01-11
JMeter y JSF. Extracción del parámetro ViewState

Últimas ofertas de empleo

2009-07-31
T. Información - Operador (dia / noche) - BARCELONA.

2009-06-25
Atención a cliente - Call Center - BARCELONA.

2009-06-19
Otras - Ingenieria (minas, puentes y puertos) - VALENCIA.

2009-06-17
Comercial - Ventas - ALICANTE.

2009-06-03
Comercial - Ventas - VIZCAYA.

view plain print ?

```java
package com.autentia.tutoriales.jsf;

import java.io.IOException;

import javax.annotation.PostConstruct;
import javax.annotation.Resource;
import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.context.FacesContext;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.context.annotation.Scope;
import org.springframework.context.support.MessageSourceAccessor;
import org.springframework.stereotype.Controller;

import com.octo.captcha.service.image.ImageCaptchaService;

@Controller
@Scope("request")
public class UserCtrl {

    private static final Log log = LogFactory.getLog(UserCtrl.class);

    private String userName;

    private String welcomeMessage;

    private String welcome;

    private Captcha captcha;

    @Resource
    private MessageSourceAccessor messageSourceAccessor;

    @Resource
    private ImageCaptchaService imageCaptchaService;

    @SuppressWarnings("unused")
    @PostConstruct
    private void init() {
        welcome = messageSourceAccessor.getMessage("home.welcome");
        captcha = new Captcha(imageCaptchaService);
    }

    /**
      * Antes de aceptar los datos del formulario se debe pasar la validación del Captcha
      *
      * @return
      */
    public String accept() {
        if (!captcha.isValidCaptcha()) {
            final String message = messageSourceAccessor.getMessage("message.captcha.error");
            addMessage(null, FacesMessage.SEVERITY_ERROR, message, message);

            newCaptcha();
            return "";
        }

        welcomeMessage = welcome + " " + userName;

        return "home";
    }

    /**
      * Genera una nueva imagen de Captcha
      *
      * @return
      */
    public String newCaptcha() {
        try {
            welcomeMessage = null;
            captcha.generateCaptchaImage();
        } catch (IOException e) {
            log.error("Error generando captcha: ' + e.getMessage());
        }
        return "home";
    }

    public static void addMessage(String clientId, Severity severity, String summary, String detail) {
        final FacesContext context = FacesContext.getCurrentInstance();

        if (context != null) {
            final FacesMessage message = new FacesMessage(severity, summary, detail);
            context.addMessage(clientId, message);
        }
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getWelcomeMessage() {
```

Por último la página home.jspx para sacar el formulario queda así:

```
view plain  print  ?
01.  < ?xml version="1.0" encoding="UTF-8"? >
02.
03.  < jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:ui="http://java.sun.com/jsf/facelets"
04.      xmlns:h="http://java.sun.com/jsf/html" xmlns:f="http://java.sun.com/jsf/core" xmlns:c="http://java.sun
05.      xmlns:ice="http://www.icesoft.com/icefaces/component">
06.
07.      < ui:composition template="/WEB-INF/facelets/template/defaultLayout.jspx" >
08.          < ui:define name="content">
09.              <h1><ice:outputText value="#{msg['home.welcome']}"/></h1>
10.
11.              <fieldset>
12.                  <legend>#{msg['home.userData']}</legend>
13.
14.                  < ice:panelGrid columns="3">
15.                      <!-- Nombre -->
16.                      < ice:outputLabel for="userName" value="#{msg['home.userNameLabel']}: "/>
17.                      < ice:inputText id="userName" value="#{userCtrl.userName}" required="true"/>
18.                      < ice:message for="userName" errorClass="error"/>
19.
20.                      <!-- Captcha -->
21.                      < ice:outputLabel for="captcha" value="#{msg['home.captcha']}: "/>
22.                      < ice:inputText id="captchaValue" value="#{userCtrl.captcha.captchaValue}' required="t
23.                      < ice:message for="captchaValue" errorClass="error"/>
24.                  < /ice:panelGrid>
25.
26.                  <!-- Etiqueta que recoge el flujo de bytes de la imagen del captcha y la pinta en pantall
27.                  < ice:graphicImage id="captcha" value="#{userCtrl.captcha.captchaImage}' />
28.                  <br/><br/>
29.
30.                  < ice:commandButton action="#{userCtrl.newCaptcha}" value="#{msg['button.newCaptcha']}' im
31.                  < ice:commandButton value="#{msg['button.accept']}" action="#{userCtrl.accept}"/><br/><br/
32.                  < ice:messages globalOnly="true" style="color:red;"/>
33.              </fieldset>
34.
35.              <br />
36.
37.              < ice:panelGroup rendered="#{not empty userCtrl.welcomeMessage}'>
38.                  <h2><strong><ice:outputText id="welcomeMessage" value="#{userCtrl.welcomeMessage}' /></str
39.              </ice:panelGroup>
40.          </ui:define>
41.      </ui:composition>
42.  </jsp:root>
```

El resultado del formulario queda así:

Si metemos mal la palabra del captcha no se registrarán los datos del formulario por lo que impedimos acceder a la lógica de negocio:

Es importante regenerar el captcha cada vez que el usuario falla ya que de lo contrario podrían romper nuestra seguridad con un ataque de diccionario. También se le suele dar al usuario la posibilidad de generar un nuevo captcha debido a que algunas veces resulta complicado averiguar la palabra que se esconde en el captcha.

Una vez que el usuario introduce la palabra correctamente pasamos la validación y llamaríamos a la lógica de negocio, en nuestro caso recogemos el nombre del usuario y le damos la bienvenida.

5. Configuración avanzada de JCaptcha

Por defecto JCaptcha utiliza un diccionario de palabras en inglés pero podemos configurar esta opción añadiendo nuestro propio diccionario asociado al locale del usuario.

Como dijimos en la introducción JCaptcha nos permite personalizar completamente la generación de la imagen a través de los servicios correspondientes en cada caso. Para insertar nuestro propio fichero de palabras debemos crear un toodlist_es (para el locale en español). También nos permite configurar los colores del texto, el tamaño de la fuente, el fondo de la imagen así como los filtros para deformar la imagen. Un ejemplo de esta configuración sería el siguiente:

view plain print ?

```
 < bean id="imageCaptchaService" class="com.octo.captcha.service.image.DefaultManageableImageCaptchaServi
    < property name="captchaEngine" ref="captchaEngine"/>
< /bean>

< bean id="captchaEngine" class="com.octo.captcha.engine.GenericCaptchaEngine'>
    < constructor-arg index="0">
        < list>
            < ref bean="CaptchaFactory"/>
        < /list>
    < /constructor-arg>
< /bean>

< bean id="CaptchaFactory" class="com.octo.captcha.image.gimpy.GimpyFactory' >
    < constructor-arg><ref bean="wordgen"/></constructor-arg>
    < constructor-arg><ref bean="wordtoimage"/></constructor-arg>
< /bean>

< bean id="wordgen" class= "com.octo.captcha.component.word.wordgenerator.DictionaryWordGenerator >
    < constructor-arg><ref bean="filedict"/></constructor-arg>
< /bean>

< bean id="filedict" class="com.octo.captcha.component.word.FileDictionary' >
    < constructor-arg index="0"><value>toddlist</value></constructor-arg>
< /bean>

< bean id="wordtoimage" class="com.octo.captcha.component.image.wordtoimage.FilteredComposedWordToImage >
    < constructor-arg index="0"><ref bean="fontGenRandom"/></constructor-arg>
    < constructor-arg index="1"><ref bean="backGenUni"/></constructor-arg>
    < constructor-arg index="2"><ref bean="simpleWhitePaster"/></constructor-arg>
    < constructor-arg index="3"><ref bean="crystal"/></constructor-arg>
    < constructor-arg index="4"><ref bean="sphere"/></constructor-arg>
    < constructor-arg index="5"><ref bean="emboss"/></constructor-arg>
< /bean>

< bean id="fontGenRandom" class="com.octo.captcha.component.image.fontgenerator.RandomFontGenerator >
    < constructor-arg index="0"><value>100</value></constructor-arg>
    < constructor-arg index="1"><value>100</value></constructor-arg>
    < constructor-arg index="2">
        < list>
            < ref bean="fontArial"/>
        < /list>
    < /constructor-arg>
< /bean>

< bean id="fontArial" class="java.awt.Font" >
    < constructor-arg index="0"><value>Arial</value></constructor-arg>
    < constructor-arg index="1"><value>0</value></constructor-arg>
    < constructor-arg index="2"><value>8</value></constructor-arg>
< /bean>

< bean id="backGenUni" class="com.octo.captcha.component.image.backgroundgenerator.UniColorBackgroundGene
    < constructor-arg index="0"><value>250</value></constructor-arg>
    < constructor-arg index="1"><value>100</value></constructor-arg>
< /bean>

< bean id="simpleWhitePaster" class="com.octo.captcha.component.image.textpaster.SimpleTextPaster >
    < constructor-arg type="java.lang.Integer" index="0"><value>3</value></constructor-arg>
    < constructor-arg type="java.lang.Integer" index="1"><value>5</value></constructor-arg>
    < constructor-arg type="java.awt.Color" index="2"><ref bean="color"/></constructor-arg>
< /bean>

< bean id="color" class="java.awt.Color" >
    < constructor-arg type="int" index="0"><value>100</value></constructor-arg>
    < constructor-arg type="int" index="1"><value>200</value></constructor-arg>
    < constructor-arg type="int" index="2"><value>56</value></constructor-arg>
< /bean>

< bean id="sphere" class="com.jhlabs.image.SphereFilter' >
    < property name="refractionIndex"><value>1</value></property>
< /bean>

< bean id="rippleBack" class="com.jhlabs.image.RippleFilter' >
    < property name="waveType"><value>3</value></property>
    < property name="XAmplitude"><value>10</value></property>
    < property name="YAmplitude"><value>3</value></property>
    < property name="XWavelength"><value>20</value></property>
    < property name="YWavelength"><value>10</value></property>
    < property name="edgeAction"><value>1</value></property>
< /bean>

< bean id="ripple3" class="com.jhlabs.image.RippleFilter' >
    < property name="waveType"><value>5</value></property>
    < property name="XAmplitude"><value>5</value></property>
    < property name="YAmplitude"><value>5</value></property>
    < property name="XWavelength"><value>10</value></property>
    < property name="YWavelength"><value>10</value></property>
    < property name="edgeAction"><value>1</value></property>
< /bean>

< bean id="emboss" class="com.jhlabs.image.EmbossFilter' >
    < property name="bumpHeight"><value>1.0</value></property>
< /bean>

< bean id="smear" class="com.jhlabs.image.SmearFilter" >
    < property name="shape"><value>0</value></property>
    < property name="distance"><value>15</value></property>
    < property name="density"><value>0.4</value></property>
    < property name="scatter"><value>0.5</value></property>
```

El resultado del captcha con esta configuración cambia totalmente:

6. Conclusiones

El resultado de la utilización de esta herramienta no puede ser más satisfactorio ya que permite de forma muy sencilla configurarla y adecuarla a nuestras necesidades gracias al soporte de Spring. Siempre es mucho mejor utilizar librerías de terceros para nuestros proyectos que tener que hacer todo nosotros. Imaginaos el esfuerzo que supondría crear algo equivalente a JCaptcha si lo necesitáramos para nuestros proyectos.

Un saludo. Juan.

- 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. 

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

Formación en nuevas tecnologías

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
JCaptcha - Generación de Captchas en Java	En este tutorial veremos la librería JCaptcha. Se trata de una librería opensource para generación de captchas en Java.	2010-04-26	3	-	-	
Instalar Puente PHP-Java en Tomcat	Hace ya algún tiempo que publicamos un tutorial de cómo se instalaba el Puente PHP-Java en un servidor web Apache. Por suerte, las versiones han cambiado para bien y ahora la instalación es mucho más sencilla.	2010-04-23	167	-	-	
Jetspeed-2 de Apache Software Foundation	Jetspeed-2 es la solución propuesta por Apache Foundation para llenar el vacío existente en cuanto a la tecnología de portales se refiere. Jetspeed-2 es un portal open source basado en J2EE y cuya principal capacidad es la creación de portlets.	2010-04-08	446	-	-	
Cómo instalar la JDK de SUN en Fedora Linux	En este tutorial vamos a ver cómo instalar la JDK de SUN. En muchos casos esto no tiene por qué ser necesario ya que Fedora ya trae una máquina virtual para Java, pero al no ser la "oficial" podemos encontrarnos con algunas incompatibilidades.	2010-03-03	1246	-	-	
PHP Vs Java	El cometido de este documento es el de realizar un análisis en profundidad de dos tecnologías ampliamente aceptadas por la comunidad diseñadora de portales web, como son PHP y Java.	2010-01-04	4078	-	-	
Procesador Inteligente de Eventos (IEP) con OpenESB	En este tutorial mostramos un ejemplo practico de gestion de eventos en SOA con IEP (Intelligent Event Processor) de OpenESB y probamos el resultado con soapUI	2010-01-04	1891	-	-	
Tutorial de BPEL con OpenESB (II)	Continuación del Tutorial de BPEL con OpenESB (I).	2009-12-29	2645	-	-	
Tutorial de BPEL con OpenESB (I)	En este tutorial vamos a aprender a crear procesos BPEL practicando con un ejemplo: un proceso de negocio de venta online de libros.	2009-12-29	3442	Muy bueno	1	
JavaBean Datasource Ireport	La particularidad del caso que nos ocupa, es conseguir que la fuente de datos del informe sea una lista de JavaBeans y no una consulta definida previamente en el informe.	2009-12-14	2592	Bueno	1	
Instalación de Glassfish 2.1	En este tutorial nos veremos cómo instalar el servidor de aplicaciones GlassFish. Además veremos los primeros pasos, como entrar en la consola de administración del servidor, y desplegar una aplicación EAR (Enterprise Application)	2009-11-11	4425	Muy bueno	2	

Nota: