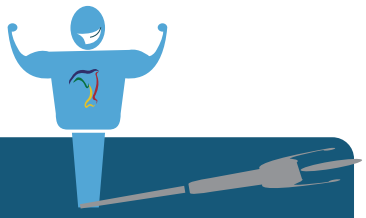
autentia

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...

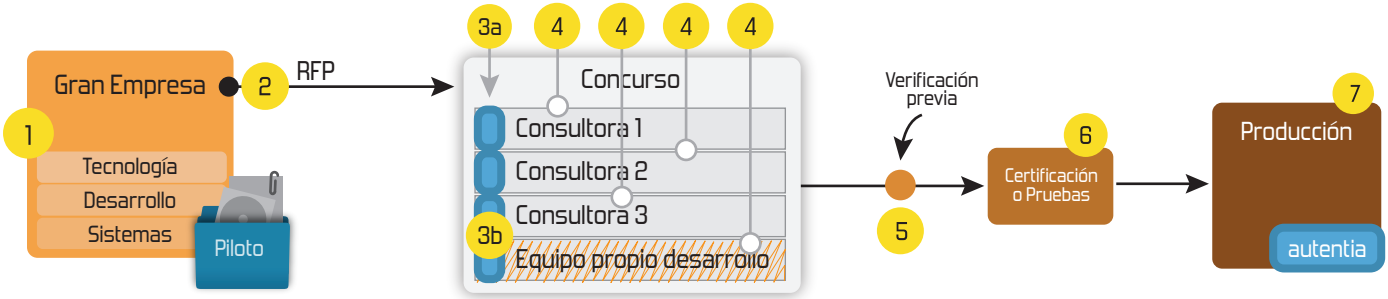
1. Desarrollo de componentes y proyectos a medida



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

DESARROLLADO POR:
Jose Manuel Sánchez Suárez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Fecha de publicación del tutorial: 2010-07-23

Envío de correo electrónico con el soporte de Jboss Seam.

0. Índice de contenidos.

1. Introducción.

Jboss Seam nos da soporte para el envío de correo electrónico, con una característica bastante interesante que radica en el hecho de que las plantillas de correo pueden generarse con facelets.

Ya hemos visto en otros tutoriales cómo enviar emails haciendo uso de las facilidades del framework de desarrollo con el que estemos trabajando, en el caso de Spring tenemos este tutorial y otros mas avanzados. Ahora es el turno de Jboss Seam y vamos a hacer uso de este tutorial para comentar otras características colaterales al propio servicio de envío de correo electrónico, que salen a la luz por la necesidad de configurar el buzón de correo en función del entorno, así veremos la posibilidad de cargar los valores de un fichero de propiedades en el entorno del fichero de configuración de Seam (components.xml) y la posibilidad de declarar un componente de Seam a través de la configuración de dicho xml inyectando ciertos valores de configuración.

El objetivo es el de disponer de un servicio propio, un componente de Seam, configurado para el envío de correo electrónico y que pueda ser inyectado en cualquier componente que requiera su uso. Mostraremos cómo crear una plantilla base para el envío de correo con el soporte de facelets de modo que establezca la estructura básica de la misma.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 17' (2.93 GHz Intel Core 2 Duo, 4GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Snow Leopard 10.6.1
- Jboss Seam 2.2.0.GA
- Maven 2.2.1.
- Eclipse 3.5: Galileo, con IAM (plugin para Maven).
- GlassFish v.2.1 con la jdk 1.6.

3. Configuración.

El soporte para el envío de email es un módulo de Seam que no se distribuye con las librerías del core, con lo que lo primero que debemos hacer es incluir la dependencia en el módulo que lo necesite:

```
<dependency>
    <groupId>org.jboss.seam</groupId>
    <artifactId>jboss-seam-mail</artifactId>
    <version>${seam.version}</version>
</dependency>
```

En nuestro caso ${seam.version} es una propiedad que tenemos definida a nivel de pom.xml para su reutilización, su valor es 2.2.0.GA.

Para habilitar el soporte de email debemos incluir lo siguiente en el fichero de configuración de Seam (components.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components"
            xmlns:core="http://jboss.com/products/seam/core"
            xmlns:international="http://jboss.com/products/seam/international"
            xmlns:security="http://jboss.com/products/seam/security"
            xmlns:web="http://jboss.com/products/seam/web"
            xmlns:transaction="http://jboss.com/products/seam/transaction"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:persistence="http://jboss.com/products/seam/persistence"
            xmlns:resteasy="http://jboss.com/products/seam/resteasy"
            xmlns:ui="http://jboss.com/products/seam/ui"
            xsi:schemaLocation=
            "http://jboss.com/products/seam/core http://jboss.com/products/seam/core-2.0.xsd
            http://jboss.com/products/seam/security http://jboss.com/products/seam/security-2.0.xsd
            http://jboss.com/products/seam/web http://jboss.com/products/seam/web-2.0.xsd
            http://jboss.com/products/seam/transaction http://jboss.com/products/seam/transaction-2.0.xsd
            http://jboss.com/products/seam/components http://jboss.com/products/seam/components-2.0.xsd
            http://jboss.com/products/seam/international http://jboss.com/products/seam/international-2.1.xsd
            http://jboss.com/products/seam/persistence http://jboss.com/products/seam/persistence-2.0.xsd
            http://jboss.com/products/seam/resteasy http://jboss.com/products/seam/resteasy-2.2.xsd"
            xmlns:mail="http://jboss.com/products/seam/mail">

   ...

   <mail:mail-session host="@mail.host@" port="@mail.port@" username="@mail.username@" password="@mail.password@" />

   ...

</components>
```

También se puede configurar, vía jndi, contra una sesión de mail configurada a nivel de servidor de aplicaciones, nosotros optamos por la vía de una configuración a nivel de aplicación puesto que el acceso a través de jndi ya nos dio problemas a la hora de configurar el servicio de autorización y autenticación (JAAS) contra GlassFish.

En los valores de los atributos de la configuración del correo lo que hacemos es introducir una serie de claves, comprendidas entre arrobas, que hacen referencia a entradas en un fichero de propiedades que Jboss Seam nos permite externalizar de modo que basta con incluirlo en el classpath del servidor (en el caso de GlassFish dentro de /glassfish/domains/domain1/lib/classes) con el nombre components.properties. Conforme a lo configurado, dicho fichero de propiedades, debería tener un contenido similar al siguiente:

Creación de servicios web RESTful con el soporte de RESTeasy de Jboss Seam.

Facebook Social Plugins

EGit un plugin de Eclipse para el sistema de control de versiones distribuido Git

Crear una estructura compleja del tipo Array en un servicio web Axis2

Últimos Tutoriales del Autor

Creación de servicios web RESTful con el soporte de RESTeasy de Jboss Seam.

Facelets en JSF 2: sistema de plantillas y componentes por composición.

DbVisualizer free version.

Session TimeOut en RichFaces, con el soporte de Jboss Seam.

Retrasar la carga de Javascript con jQuery.getScript().

Síguenos a través de:

Últimas ofertas de empleo

2010-06-25
T. Información - Analista / Programador - BARCELONA.

```
# Sets the service as active or not
mail.active=true
# SMTP server used to send emails
mail.host=smtp.host.com
# Port used by the SMTP server to send emails (default port is 25)
mail.port=25
# user authentication in the mail.host
mail.username=info@autentia.es
# password for the user authentication
mail.password=password

# Email address as notifier for all the emails sent
app.mail.notifierMailAddress=info@autentia.com

# Name of the notifier for all the emails sent
app.mail.notifierName=Dpto. administrativo de Autentia
```

La primera y las dos últimas entradas del fichero las usaremos en el siguiente punto.

4. Servicio de envío de email.

Antes de mostrar la clase que implementará el servicio, vamos a apoyarnos en una clase que encapsulará las características típicas de emisor y receptor del correo electrónico. Con ello en la plantilla mostraremos la posibilidad de acceder a objetos complejos.

```
package com.autentia.commons.mail;

import java.io.Serializable;

/**
 * Mail User, to use in the mail service.
 *
 * @author Autentia Real Business Solutions S.L.
 * @see http://www.autentia.com
 *
 */
public class MailUser implements Serializable {

    private static final long serialVersionUID = -3811718498179974422L;

    public MailUser(){
    }

    public MailUser(String name, String email){
        this.name = name;
        this.email = email;
    }

    /** the name */
    private String name;

    /** the email */
    private String email;

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

Ahora sí, el contenido de la clase de implementación del servicio podría tener un contenido como el que sigue:

```
package com.autentia.commons.mail;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;

import org.jboss.seam.annotations.In;
```

```
08 import org.jboss.seam.contexts.Contexts;
09 import org.jboss.seam.faces.Renderer;
10 import org.jboss.seam.web.ServletContexts;
11 import org.slf4j.Logger;
12 import org.slf4j.LoggerFactory;
13
14 /**
15  * Mail Service
16  *
17  * @author Autentia Real Business Solutions S.L.
18  * @see http://www.autentia.com
19  *
20  */
21 public class MailService implements Serializable {
22
23     private static Logger log = LoggerFactory.getLogger(MailService.class);
24
25     private static final long serialVersionUID = 7293359358555947008L;
26
27     private static final String PATH_PREFIX = "/emails/";
28
29     private static final String PATH_SUFFIX = ".xhtml";
30
31     private boolean active = true;
32
33     @In(required=false)
34     private Renderer renderer;
35
36     /** the notifier */
37     private MailUser from = new MailUser();
38
39     // seam does nor support Compound property names, like mailService.from.name
40     // need for the components.xml injection
41     public void setFromName(String name) {
42         this.from.setName(name);
43     }
44
45     public void setFromEmail(String email) {
46         this.from.setEmail(email);
47     }
48
49     public MailUser getFrom() {
50         return from;
51     }
52
53     public void setFrom(MailUser from) {
54         this.from = from;
55     }
56
57     public boolean isActive() {
58     return active;
59     }
60
61     public void setActive(boolean active) {
62     this.active = active;
63     }
64
65     public void send(String template, MailUser to){
66         send(template, to, null);
67
68     }
69
70     public void send(String template, MailUser to, Map<String,Object> params){
71         if (!active){
72             log.trace("Mail service is inactive.");
73             return;
74         }
75
76         if (params == null){
77             params = new HashMap<String,Object>();
78         }
79
80         params.put("to",to);
81         params.put("from", from);
82
83         for (Map.Entry<String, Object> entry : params.entrySet()) {
84             Contexts.getSessionContext().set(entry.getKey(), entry.getValue());
85         }
86
87         renderer.render(PATH_PREFIX + template + PATH_SUFFIX);
88     }
89 }
```

Tiene las siguientes características:

- tres propiedades que nos van a permitir asignar:

    - si el servicio está activo o no, ideal para marcarlo como inactivo en el entorno de tests,
    - el nombre y el correo electrónico del remitente, que serán comunes para todos los emails enviados a través de este servicio. En este punto, aún teniendo inicializado el objeto from de tipo MailUser, vía inyección de dependencias no podemos inyectar valores en atributos compuestos como con Spring, de ahí la necesidad de los métodos setFromEmail y setFromName.
- dos métodos para realizar el envío del correo electrónico (uno con parámetros y otro no) que reciben el nombre de la plantilla que deben cargar para componer el cuerpo y las características del mensaje y el destinatario del correo encapsulado en un objeto de tipo MailUser.
- la asignación de las propiedades del correo se realiza a través de la inserción de las mismas en el contexto de sesión y el envío se realiza mediante un renderizador. Esto último es lo más interesante puesto que la plantilla es facelets y lo que se ejecuta es la última fase del ciclo de vida de JSF, la de renderización, para el árbol de componentes que contiene la plantilla. Dentro de la plantilla se podrán tener referencias a las propiedades asignadas a través del contexto o incluso a aquellos componentes de Seam con un ámbito de sesión.
- la plantilla de facelets, el render la obtiene del classpath con un prefijo y sufijo predeterminados.

La clase no está anotada para que se incluya de forma automática como un componente de Seam porque requiere de ciertas propiedades que obtenemos del fichero de configuración de entorno (components.xml) antes expuesto y, para ello, tenemos que obtenerlas mediante las claves indicadas en el punto anterior.

Para publicar el servicio debemos incluir lo siguiente en el fichero de configuración de Seam (components.xml):

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <components ...>
03
04     ...
05
06
07     <component name="mailService" class="com.autentia.commons.mail.MailService" auto-
   create="true">
08         <property name="active">@mail.active@</property>
09         <property name="fromEmail">@app.mail.notifierMailAddress@</property>
10         <property name="fromName">@app.mail.notifierName@</property>
11     </component>
12
13     ...
14
15 </components>
```

Con ello, todas las propiedades relativas a la configuración del servicio de correo quedan externalizadas en el fichero de propiedades (components.properties) y, de este modo, en el entorno de tests podemos hacer uso de otro fichero de propiedades que deshabilite el servicio de correo para evitar el envío innecesario de emails en la ejecución de las pruebas.

5. Ejemplo de uso del servicio.

A continuación vamos a mostrar un ejemplo de uso del servicio, en el que un usuario puede solicitar el refresco de su contraseña tras haberla olvidado.

```
01 import org.jboss.seam.annotations.In;
02 import org.jboss.seam.international.StatusMessages;
03 import org.jboss.seam.international.StatusMessage.Severity;
04 import org.jboss.seam.util.RandomStringUtils;
05
06 import com.autentia.commons.mail.MailService;
07 import com.autentia.commons.mail.MailUser;
08
09 import com.autentia.core.entities.User;
10
11
12 @AutoCreate
13 @Name("homeBean")
14 @Scope(ScopeType.SESSION)
15 public class HomeBean implements Serializable {
16
17     ...
18
19     @In
20     private MailService mailService;
21
22     @In
23     private StatusMessages statusMessages;
24
25     private String generatedPassword;
26
27     ...
```

```
28
29      public String forgotPassword() {
30
31          final User user = getUserAunthenticated();
32
33          generatedPassword = RandomStringUtils.randomAlphanumeric
    (User.PASSWORD_LENGTH);
34
35          mailService.send("forgotPassword", new MailUser(user.getName(), user.getEmail
    ()));
36
37          statusMessages.addFromResourceBundle(Severity.INFO, "info.passwordSent");
38
39      }
40
41      public String getGeneratedPassword() {
42          return generatedPassword;
43      }
44
45      ...
46
47      private User getUserAunthenticated(){
48         ...
49      }
50
51  }
```

A destacar lo siguiente:

- 19: la inyección del servicio de envío de email,
- 21: la inyección del contexto de mensajes de notificación al usuario en la respuesta,
- 33: la clase de utilidades de Seam RandomStringUtils que nos permite generar una cadena con caracteres aleatorios en función de una longitud dada,
- 35: la invocación al servicio de envío pasándole el nombre de la plantilla y la información del usuario conectado a la aplicación.
- 37: el hecho de añadir un mensaje al contexto para informar del envío correcto del email.

A continuación vemos el contenido de la plantilla de email relacionada con el olvido de la contraseña en la que nos limitamos a definir el contenido de subject y del cuerpo del mensaje, basándonos en el soporte a la internacionalización de JSF y en la referencia a las propiedades asignadas por el servicio e incluso en propiedades del controlador que realiza el envío (como la nueva contraseña generada).

```
01  <ui:composition xmlns:h="http://java.sun.com/jsf/html"
02                   xmlns:ui="http://java.sun.com/jsf/facelets"
03                   template="/WEB-INF/facelets/template/email.jspx">
04
05      <ui:define name="subject">
06          <h:outputText value="#{messages
    ['forgotPasswordTemplate.changePasswordRequest']}" />
07      </ui:define>
08
09      <ui:define name="body">
10          <p><h:outputText value="#{messages
    ['forgotPasswordTemplate.receivedRequest']}" /></p>
11          <p><h:outputText value="#{messages
    ['forgotPasswordTemplate.userData']}" /></p>
12          <p><h:outputText value="#{messages['user']}" />: <strong><h:outputText
    value="#{to.email}" /></strong></p>
13          <p><h:outputText value="#{messages['password']}" />: <strong><h:outputText
    value="#{homeBean.generatedPassword}" /></strong></p>
14          <br />
15          <p><h:outputText value="#{messages['forgotPasswordTemplate.footer']}" /></p>
16      </ui:define>
17
18  </UI:COMPOSITION>
```

Lo más interesante es que la plantilla, a su vez, hace uso de una plantilla de facelets (template="/WEB-INF/facelets/template/email.jspx"), cuyo contenido veremos después.

El texto internacionalizado reside en ficheros de propiedades que hacen referencia al locale en su nombre y que deberían tener un contenido similar al siguiente:

```
1  #forgot password template
2  forgotPasswordTemplate.changePasswordRequest=Petici\u00F3n de cambio de
   contrase\u00F1a
3  forgotPasswordTemplate.receivedRequest=Hemos recibido una petici\u00F3n de cambio de
   contrase\u00F1a.
4  forgotPasswordTemplate.userData=Estos son sus datos de acceso a la aplicaci\u00F3n
5  forgotPasswordTemplate.footer=Podr\u00E1 cambiar su password una vez autenticado en
   el sistema desde la opci\u00F3n 'Cambiar contrase\u00F1a' del men\u00FA superior.
```

Por último, el contenido de la plantilla maestra para todas las plantillas de email debería tener un contenido como el que sique:

```
01  <m:message xmlns="http://www.w3.org/1999/xhtml"
```

```
        xmlns:m="http://jboss.com/products/seam/mail"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:ui="http://java.sun.com/jsf/facelets">

    <m:from name="#{from.name}" address="#{from.email}" />
    <m:to name="#{to.name}">#{to.email}</m:to>
    <m:subject><ui:insert name="subject" /></m:subject>

    <m:body>
    <html>
        <body>
        <ui:insert name="body" />
        </body>
    </html>
    </m:body>
</m:message>
```

Son componentes propios del módulo de envío de email que hacen referencia a las propiedades asignadas a nivel de contexto para el envío, y realizan la inserción del valor de variables declaradas en la plantilla que hace uso de la misma (el subject y el cuerpo del mensaje).

6. Asignación de la url del contexto de la aplicación como parámetro a la plantilla.

Para quienes trabajamos con JSF y facelets, el hecho de disponer de dicha tecnología a la hora de componer el contenido de un mensaje, nos podemos ver tentados a hacer uso de una expresión como la que sigue para obtener la url y el contexto de la aplicación:

```
<a href="#
{facesContext.externalContext.requestContextPath}/external/deliveryTracking">
    <h:outputText value="#
{facesContext.externalContext.requestContextPath}/external/deliveryTracking" />
</a>
```

Quizás lo necesitemos para inluir un link a la aplicación o una imagen.

Podría funcionar pero no, lo que imprime es siempre "project", es un valor fijo. El contexto en el que se renderiza la plantilla de correo no es el de la petición del cliente con lo que no tenemos acceso a dicha información.

Podemos solventar el problema asignando una propiedad adicional al contexto de ejecución de la plantilla, como sique:

```
params.put("baseUrl", ServletContexts.getInstance().getRequest().getScheme() + "://"
+
                      ServletContexts.getInstance().getRequest().getServerName() +
":" +
                      ServletContexts.getInstance().getRequest().getServerPort() +
                      ServletContexts.getInstance().getRequest().getContextPath());
```

De este modo, desde la plantilla podemos hacer referencia al mismo, como lo hacemos con el resto de propiedades.

```
<a href="#{baseUrl}/external/deliveryTracking">
    <h:outputText value="#{baseUrl}/external/deliveryTracking" />
</a>
```

7. Referencias.

- http://docs.jboss.org/seam/2.2.0.GA/reference/en-US/html/mail.html

8. Conclusiones.

En este tutorial hemos analizado temas algo más avanzados de los que venimos tratando, relacionados con Jboss Seam.

Ya llevamos un tiempo trabajando con Seam, el suficiente para que ahora, con un cambio de versión, se vengan abajo los conocimientos consolidados... :-D, tenemos pendiente analizar la versión 3 (alpha aún) del framework que será, en su núcleo (WELD), la implementación de referencia del nuevo contenedor de inversión de control de JEE6.

Si estáis interesados en el contenido de nuestros tutoriales y tenéis una necesidad formativa al respecto no dudeis en poneros en contacto con nosotros. En Autentia nos dedicamos, además de a la consultoría, desarrollo y soporte a desarrollo, a impartir cursos de formación de las tecnologías con las que trabajamos.

Un saludo.

Jose

jmsanchez@autentia.com

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.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS