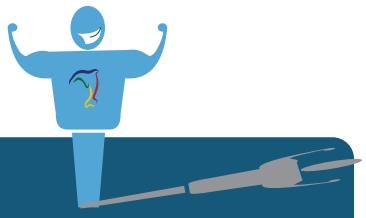
autentia

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...

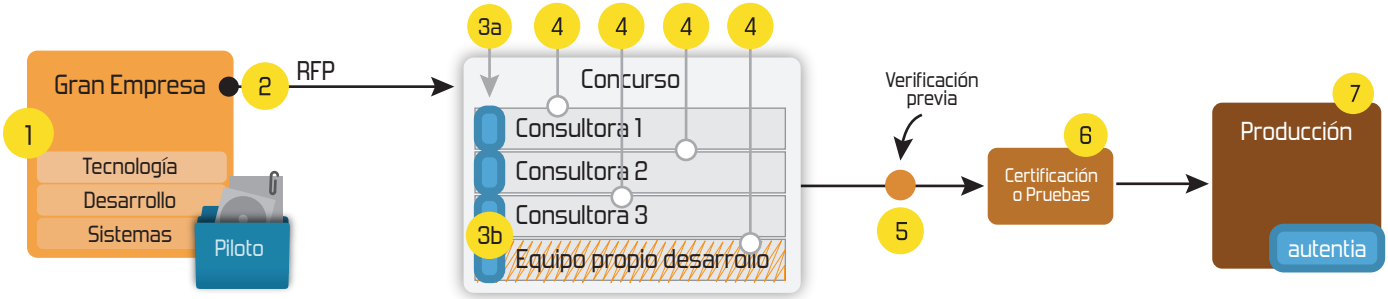
1. Desarrollo de componentes y proyectos a medida



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: 2010-09-04

Gestión de eventos en el cliente con el soporte de Ajax de RichFaces.

0. Índice de contenidos.

1. Introducción

En este tutorial vamos a proponer una solución sencilla a la necesidad de provocar un evento en el cliente cuando termine el procesamiento de una petición en el servidor, con el soporte de Ajax (A4JSF) de RichFaces.

Como digo, la solución es sencilla, si bien, después vamos a complicarla condicionando el evento en el cliente al procesamiento correcto o no de la petición en el servidor.

Para más información sobre RichFaces podéis consultar el resto de tutoriales que tenemos en adictos, un buen comienzo puede ser el de Introducción a RichFaces... ;-)

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.
- RichFaces 3.3.3.CR1
- Maven 2.2.1.
- Eclipse 3.5: Ganymede, con IAM (plugin para Maven).
- Apache Tomcat 6.0.20 con la jdk 1.5.

3. El atributo onComplete.

El objetivo es renderizar en el cliente un botón que al pulsarse se deshablite y provoque un evento vía Ajax en el servidor, tras la respuesta del servidor mostrará un mensaje al cliente, una alerta para simplificar el ejemplo, y se volverá a habilitar.

La solución es la siguiente:

```
01 <a4j:form id="reportForm">
02
03 <h:panelGrid columns="3">
04   <h:outputLabel id="labelForReference" for="reference"
   value="Referencia">
05
06   <h:inputText id="reference" value="#{productReportCtrl.reference}"
   required="true">
07     <a4j:support event="onblur" reRender="generate" />
08   </h:inputText>
09
10   <a4j:commandButton id="generate" value="#{messages
   ['action.generateReport']}"
11     action="#{productReportCtrl.generate}"
12     oncomplete="alert('Informe generado
   correctamente');this.disabled=false;"
13     onclick="this.disabled=true;" />
14
15   </h:panelGrid>
16
17 </a4j:form>
```

Ya había dicho que la solución era sencilla, basta con añadir al componente de acción el atributo onComplete y programar el evento que se ejecutará cuando termine el procesamiento de la petición.

Vamos a leer algo más el código:

- línea 1: incluimos un componente de formulario con el soporte de ajax,
- línea 3: incluimos un componente que renderiza una tabla con tres columnas,
- línea 5: incluimos un componente de etiqueta que pinta un texto para indicar el contenido del siguiente componente,
- línea 7 y siguientes: añadimos un componente de entrada de texto para incluir la referencia a un producto (por ejemplo), parámetro que recibirá el managedBean cuando reciba la petición,
- línea 11 y siguientes: el componente de acción que provoca la invocación al método generate del managedBean productReportCtrl, quien en función de la referencia del producto realizará la lógica de negocio necesaria,
- línea 16: cerramos el componente que renderiza la tabla,
- línea 18: cerramos el componente que renderiza el formulario,

Histórico de NOTICIAS

Últimos Tutoriales

Contratos Ágiles y TDD

SQL Joins explicados de forma gráfica con diagramas Venn

Introducción básica a la herramienta DBSchema

MediaWiki - NamespacePermissions

Prey, localizador de dispositivos móviles robados

Últimos Tutoriales del Autor

Envío de correo electrónico con el soporte de Jboss Seam.

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.

Síguenos a través de:

4. Condicionando el evento.

Como os comentaba en la introducción vamos a complicarlo algo más, condicionando el evento en el cliente al procesamiento correcto o no de la petición en el servidor.

Este sería el código del correspondiente managedBean, con el soporte de anotaciones de Jboss Seam:

```
01 @Name("productReportCtrl")
02 @Scope(ScopeType.SESSION)
03 public class ProductReportCtrl {
04
05     @In
06     protected StatusMessages statusMessages;
07
08     private String reference;
09
10     public String getReference(){
11       return reference;
12     }
13     public void setReference(String reference){
14       this.reference=reference;
15     }
16     public void generate(){
17         if (reference == notFound()){
18             statusMessages.addFromResourceBundle(Severity.ERROR,
"action.msg.error");
19             return;
20         }
21         // TODO: invocar a la lógica de negocio que genera el
informe
22     }
23
24     private boolean notFound(){
25         // TODO: invocar la lógica de negocio que comprueba que la
referencia existe
26     }
27 }
```

Y ahí va el código del xhtml:

```
01 <a4j:form id="reportForm">
02
03 <h:panelGrid columns="3">
04   <h:outputLabel id="labelForReference" for="reference"
value="Referencia">
05
06   <h:inputText id="reference" value="#{productReportCtrl.reference}"
required="true">
07     <a4j:support event="onblur" reRender="generate" />
08   </h:inputText>
09
10   <a4j:commandButton id="generate" value="#{messages
['action.generateReport']}"
11     action="#{productReportCtrl.generate}"
12     data="#{facesContext.maximumSeverity.ordinal ge 2}"
13     oncomplete="if(data == false) { alert('Informe generado
correctamente'); } else { alert('Se ha producido un error en la
generación') } " />
14
15   </h:panelGrid>
16
17 </a4j:form>
```

Hemos introducido en el componente de acción el atributo **data** que puede recoger un valor de cualquier managedBean o de cualquier variable de contexto publicada en el entorno de renderización de la página. El atributo acepta notación JSON, pero a nosotros solo nos interesa asignarle el resultado de una condición que evalúa si existe algún error en el contexto de JSF cuya criticidad sea mayor que WARNING (**#{facesContext.maximumSeverity.ordinal ge 2}**: el atributo maximumSeverity de la clase FacesContext almacena la mayor criticidad de los mensajes encolados con una referencia a un item de la clase Severity, nosotros hacemos uso del ordinal de la enumeración).

Con ello en el atributo onComplete podemos añadir una condición en base al contenido de ese **data** que nos permita mostrar un mensaje u otro, para el objetivo del tutorial basta, pero podríamos hacer uso de un componente de ventana modal que pintase los errores que encolamos en el managedBean y mostrarlo o no mostrarlo.

5. Referencias.

- Introducción a RichFaces
- http://livedemo.exadel.com/richfaces-demo/richfaces/commandButton.jsf?tab=info&cid=250770
- http://seamframework.org/Community/SeamJQueryIntegration
- http://www.icefaces.org/JForum/posts/list/9799.page

6. Conclusiones.

Nos resulta bastante útil tener este nivel de control sobre los eventos y respuestas frente a estos eventos que nos proporcionan las librerías de componentes JSF con las que trabajamos día a día.

Si os veis en la necesidad en algún momento, espero que os haya servido de ayuda.

Un saludo.

Jose

jmsanchez@autentia.com

COMENTARIOS


W3C XHTML 1.0
W3C CSS
XML RSS
XML ATOM