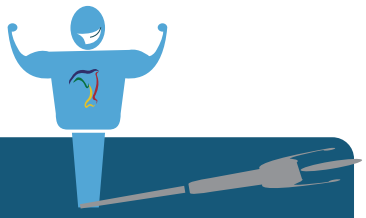
autentia

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...

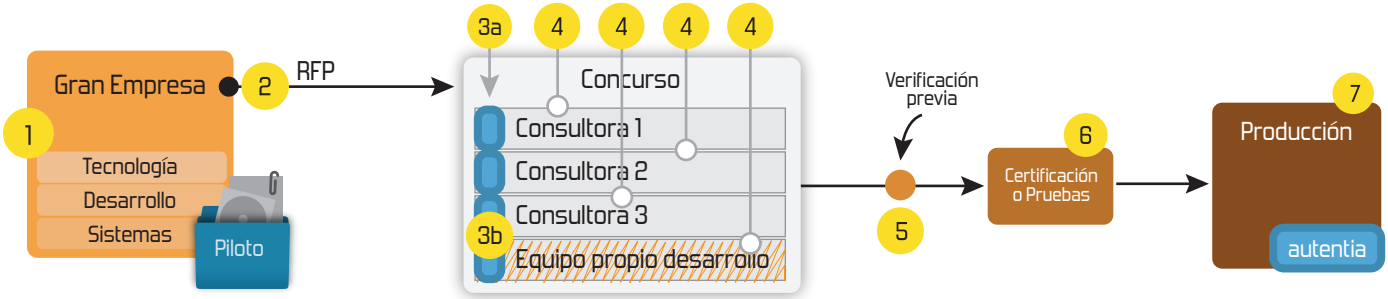
1. Desarrollo de componentes y proyectos a medida



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:
Alejandro Pérez García

Alejandro es socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Ingeniero en Informática y Certified ScrumMaster

Si te gusta lo que ves, puedes contratarle para darte ayuda con soporte experto, impartir **cursos presenciales** en tu empresa o para que **realicemos tus proyectos como factoría** (Madrid). Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Fecha de publicación del tutorial: 2011-01-26

Cosas que no funcionan en JSF2 como uno podría esperar (es decir, bugs)

Creación: 24-1-2011

Índice de contenidos

1. Introducción

En este tutorial vamos a ver algunos de los errores que hay actualmente en JSF2. Son cositas que no funcionan como uno podría esperar y que nos pueden dar más de un quebradero de cabeza, de hecho algunos son bugs reconocidos, que esperemos que arreglen algún día ;)

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 17' (2.8 GHz Intel i7, 8GB DDR3 SDRAM, 256GB Solid State Drive).
- NVIDIA GeForce GT 330M with 512MB
- Sistema Operativo: Mac OS X Snow Leopard 10.6.6

- JDK 1.6.0_22
- JSF2 2.1.0-b10 (la implementación com.sun.faces.jsf-impl)

3. c:forEach y el scope view

```
<h:form>
    <c:forEach var="item" items="#{backBeanInViewScope.itemList}" >
        <h:outputText value="#{item.label}" />
    </c:forEach>

    <h:commandLink value="Petición Ajax" actionListener="#
{backBeanInViewScope.actionListener}" >
        <f:ajax render="mensaje" />
    </h:commandLink>
    <h:outputText id="mensaje" value="Hola Mundo" />
</h:form>
```

Esto puede parecer que funciona pero lamentablemente no lo hace correctamente.

Estamos hablando que tenemos un back bean en **scope view**, esto significa que el mismo back bean se mantiene en el servidor mientras no naveguemos a otra página. En el ejemplo se puede ver como tenemos un enlace que por Ajax repinta un mensaje (el ejemplo en si es absurdo porque el mensaje es fijo, pero la gracia es hacer una petición Ajax). Si ponemos trazas en el constructor de `backBeanInViewScope` veremos que con cada petición Ajax se vuelve a crear, por lo que no se mantiene la idea de view scope. Esto se debe a un problema en el `c:forEach`, que parece que no puede acceder a este ámbito y por lo tanto crea el objeto cada vez.

Soluciones:

- Meter el back bean en sesion, cosa que no considero una opción.
- Dejar de usar el `c:forEach`, que sería la mejor opción siempre. En general todas la estiquetas `c:xxx` suelen dar problemas porque el ciclo de vida es distinto que el del resto de componentes, las etiquetas `c:xxx` sólo se ejecutan al construir la página (árbol de componentes) la primera vez, mientras que el resto de etiquetas se ejecutan cada vez que se pinta la página.

En este caso la alternativa al `c:forEach` es usar el `ui:repeat`.

4. f:ajax y renderizar un componente fuera del ui:repeat

```
<h:form id="elFormulario">
    <ui:repeat var="item" value="#{backBeanInViewScope.itemList}" >
        <h:commandLink value="Petición Ajax" actionListener="#
{backBeanInViewScope.actionListener}" >
            <f:ajax render="mensaje" />
        </h:commandLink>
    </ui:repeat>
    <h:outputText id="mensaje" value="Hola Mundo" />
</h:hform>
```

Con este ejemplo veremos que al renderizar la página se queja porque dice que "mensaje" no es un identificador válido para el render del `f:ajax`. Es porque el `f:ajax` está dentro del `ui:repeat` y el componente que queremos repintar esta fuera.

Soluciones:

- Meter el `h:outputText` dentro del `ui:repeat` funcionará correctamente (pero claro nos pintará el texto n veces). Con lo que no parece una solución muy buena.
- Identificar el componente por su nombre completo "`:elFormulario:mensaje`". Esto suele ser lo más fácil aunque no siempre es posible porque no siempre tenemos acceso al id del formulario, por ejemplo si estamos haciendo un componente por composición que no sabemos donde se va a usar.

Podríamos pensar que una solución es poner el `f:ajax` rodeando todo el `ui:repeat`. A parte de que esto sería matar moscas a cañonazos, ya que estamos dando el comportamiento Ajax a todos los componentes, además es que no funciona !!! Es decir da igual que el `f:ajax` este dentro del `ui:repeat` o rodeando el `ui:repeat`, en cuanto hay un `ui:repeat` implicado, o el componente que ponemos en el render del `f:ajax` está dentro del `ui:repeat` o tendremos que poner el nombre completo (incluyendo el nombre del formulario).

5. ui:repeat y un componente con binding

```
<h:form id="elFormulario">
    <ui:repeat var="link" value="#{backBeanInViewScope.linksList}" >
        <h:commandLink binding="#{link}" />
    </ui:repeat>
</h:hform>
```

Una de las principales características de JSF es que trabajamos con componentes que tienen una representación mediante una clase Java. Gracias a esto podemos, en nuestro back bean, crear programaticamente (hacemos un `new`) estos componentes, incializarlos como queramos y luego hacer un binding en el `xhtml` para mostrarlos. Esto permite hacer interfaces más dinámicas ya que podemos añadir componentes que antes no existían en el árbol de la página.

En el ejemplo presentamos un ejemplo donde tenemos una lista de enlaces que hemos creado dinámicamente en el back bean. Con el `ui:repeat` iteramos por esta lista para pintarlos. Bueno, pues lamentablemente esto no funciona debido a un bug. Lo curioso es que si usáis la etiqueta `ui:debug` y examináis el árbol de componentes, los enlaces están en el árbol de componentes, y si miráis el html generado, los enlaces están, pero no se muestra el "value" donde está el texto que se debería mostrar, por lo que estos enlaces quedan inaccesibles.

Soluciones:

- Usar un `c:forEach`, que si funciona. Lo malo de esto es que ya hemos dicho antes el `c:forEach` tiene bastantes problemas y limitaciones, por lo que habrá que mirar esta solución con mucho cariño. Por ejemplo si queremos un componente Ajax en nuestro enlace, ya hemos visto antes que con un `c:forEach` es imposible.
- Guardar en una clase los atributos del `c:commandLink` y en vez de poner el binding hacer referencia con EL a estos atributos, por ejemplo:

```
<h:form id="elFormulario">
    <ui:repeat var="linkAttributes" value="#
{backBeanInViewScope.linkAttributesList}" >
        <h:commandLink value="#{linkAttributes.value}" />
    </ui:repeat>
</h:hform>
```

6. Conclusiones

Todo el software tiene algún fallito, por eso es importante conocer bien lo que tenemos entre manos, tanto sus virtudes como sus defectos, para poder evitarlos y así aprovecharnos sólo de las cosas buenas ;).

Así que cuando os pasen estas cosas no os desaniméis, buscar por internet, hacer pruebas y acotar el problema en un ejemplo muy sencillo, determinar la causa, arreglarla si es posible, y sino esquivarla.

Y recordar que a las malas nos tenéis en Autentia para echaros una mano.

7. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software) y Certified ScrumMaster

Socio fundador de Autentia (Desarrollo de software, Consultoría, Formación)

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

http://www.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