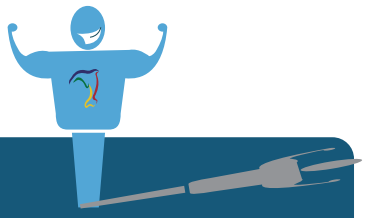
autentia

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...

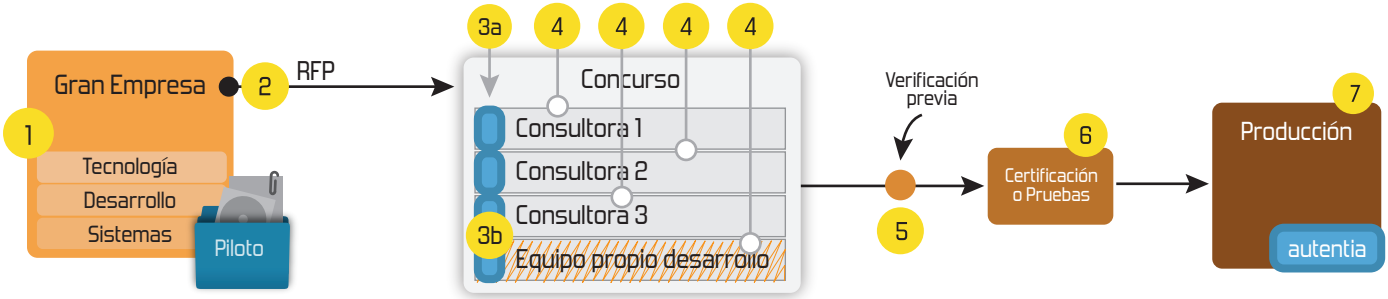
1. Desarrollo de componentes y proyectos a medida



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

» Competición Plasma Cars (Autos Locos)
» Probando con Marick
» Autentia estuvo en el Spring 2GX Day
» No todo es trabajar...
» Cambio de fecha charla Hibernate
» ¡¡¡Vuelven las Charlas de Autentia!!!
» Nuestros tutoriales alcanzan la cifra de 10 millones de visitas!!!
» Publicado el primer libro de TDD en castellano de Carlos Blé

+Noticias Destacadas

» Competición Plasma Cars (Autos Locos)
» Probando con Marick
» Autentia estuvo en el Spring 2GX Day
» ¡¡¡Vuelven las Charlas de Autentia!!!

+Comentarios Cómic

+Enlaces

Catálogo de servicios Autentia

Tríptico (6,3 MB)

Cómic (7,4 MB)

Tutorial 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 impartir **cursos presenciales** en tu empresa o para **ayudarte en proyectos** (Madrid). Puedes encontrarme en Autentia

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

Acceso de usuarios registrados:

E-mail:

Contraseña:

Entrar

Deseo registrarme

He olvidado mis datos de acceso

Registra tu empresa:

Descubre las ventajas de registrar tu empresa en AdictosAlTrabajo...

Registrar mi empresa

Listado de empresas ya registradas

Descargar este documento en formato PDF: jsf2Return.pdf

Fecha de creación del tutorial: 2010-03-09

JSF 2 ya está aquí !!!
The JSF Return, ahora más sencillo que nunca !!!

Creación: 28-02-2010

Quiero agradecer a mi compañero Germán (http://www.adictosaltrabajo.com/tutoriales-autor.php?autor=17) su ayuda para resolver varias de las cuestiones que aquí se tratan.

Índice de contenidos

Google

Web

www.adictosaltrabajo.c

Buscar

Últimos tutoriales

2010-03-09
The JSF 2 Return

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

1. Introducción

He titulado este tutorial como "JSF 2 ya está aquí !!!" aunque realmente lleva ya cierto tiempo entre nosotros. Lo que si es verdad es que es nuestro primer tutorial al respecto, y esto es porque parece que tiene madurez suficiente como para plantearse usar JSF2 en proyectos reales (actualmente están con la 2.0.2 y ya están pensando en la 2.1)

Dicen que segundas partes nunca fueron buenas. Pues en este caso no aplica, ya que JSF 2 viene con más fuerza que nunca, simplificando mucho el desarrollo y añadiendo funcionalidades nuevas. En este tutorial veremos algunas de las principales novedades, así que agarraos bien a vuestros asientos y disfrutar del viaje ;)

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, 128GB Solid State Drive).
- NVIDIA GeForce 9400M + 9600M GT with 512MB
- Sistema Operativo: Mac OS X Snow Leopard 10.6.1
- JDK 1.6.0_17
- Maven 2.2.1
- JSF 2 (2.0.2)
- Tomcat 6.0.24

3. "Instalando" JSF 2 en nuestro proyecto

El tutorial lo he planteado como un pequeño proyecto donde veremos en funcionamiento varias de las nuevas funcionalidades de JSF 2. El código completo del proyecto lo podéis descargar **aquí**

El proyecto básicamente tiene dos pantallas, una de login (más falsa que un duro de madera) y otra donde podremos dar de alta tutoriales y veremos el listado con lo que vamos añadiendo.

El proyecto está preparado con Maven, donde nos encontraremos la siguiente jerarquía de proyectos:

- autentiaJsf2-parent: Es el proyecto padre que dirige la compilación del resto de proyectos.
    - autentiaJsf2-model: Tendremos las clases del modelo.
    - autentiaJsf2-web: Tendremos las clases de la interfaz Web, y por supuesto toda la chicha sobre JSF 2 que nos interesa en este tutorial.

A lo largo del tutorial vamos a ir mostrando trozos de código y comentando las nuevas funcionalidades.

Empezaremos por enseñar como declara las dependencias de Maven para poder usar JSF 2. En el pom.xml del parent y el model no hay nada destacable, lo interesante está en el proyecto web:

```
...
<dependencies>
    <dependency>
        <groupId>com.autentia.tutorial.jsf2</groupId>
        <artifactId>autentiaJsf2-model</artifactId>
        <version>${version}</version>
    </dependency>

    <!-- API e implementación de JSF 2. Del repo dev.java.net -->
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-api</artifactId>
        <version>2.0.2-b10</version>
    </dependency>
    <dependency>
        <groupId>com.sun.faces</groupId>
        <artifactId>jsf-impl</artifactId>
        <version>2.0.2-b10</version>
    </dependency>

    <!-- API e implementación de JSR-303 (validaciones). Del repo de jboss -->
    <dependency>
        <groupId>javax.validation</groupId>
        <artifactId>validation-api</artifactId>
```

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

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

```
25          <version>1.0.0.GA</version>
26      </dependency>
27      <dependency>
28          <groupId>org.hibernate</groupId>
29          <artifactId>hibernate-validator</artifactId>
30          <version>4.0.2.GA</version>
31      </dependency>
32  </dependencies>
33
34  <repositories>
35      <repository>
36          <id>maven2-repository.dev.java.net</id>
37          <name>Java.net Repository for Maven</name>
38          <url>http://download.java.net/maven/2</url>
39      </repository>
40      <repository>
41          <id>maven2-repository.jboss.com</id>
42          <name>Jboss Repository for Maven</name>
43          <url>http://repository.jboss.com/maven2</url>
44      </repository>
45  </repositories>
46  ...
```

Podemos ver en las líneas 34-45, como se declaran dos repositorios. Esto es necesario porque los jar todavía no están disponibles en el repositorio público de ibiblio. Más arriba vemos como se declaran las dependencias: por un lado incluimos tanto el api (paquetes `javax.faces`) como la implementación (paquetes `com.sun.faces`); ambos son necesarios. También se puede ver como se incluye el `validation-api` e `hibernate-validator`. El primero es la interfaz del JSR-303 y el segundo es la implementación que da Hibernate a este JSR. Este JSR-303 ya está dentro de JSF 2 y nos permite, mediante anotaciones, especificar que validaciones queremos hacer sobre nuestros beans (más adelante veremos ejemplos de esto).

3.1. web.xml

El web.xml es extremadamente sencillo:

```
01  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance "
    xmlns="http://java.sun.com/xml/ns/javaee "
02      xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd "
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
03  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd "
04      id="WebApp_ID" version="2.5">
05
06      <display-name>Autentia JSF 2 Tutorial</display-name>
07
08      <context-param>
09          <description>
10              Define the value returned by Application.getProjectStage(). Allowed values:
    Production, Development,
11              UnitTest, SystemTest, Extension. Default value is Production.
12          </description>
13          <param-name>javax.faces.PROJECT_STAGE</param-name>
14          <param-value>Development</param-value>
15      </context-param>
16
17      <!-- Definir este parámetro es más cómodo y más visual que usar el <ui:remove> de Facelets.
    -->
18      <context-param>
19          <description>Do not render comments in facelets (xhtml) pages. Default is
    false.</description>
20          <param-name>javax.faces.FACELETS_SKIP_COMMENTS</param-name>
21          <param-value>true</param-value>
22      </context-param>
23
24      <servlet>
25          <servlet-name>Faces Servlet</servlet-name>
26          <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
27          <load-on-startup>1</load-on-startup>
28      </servlet>
29
30      <servlet-mapping>
31          <servlet-name>Faces Servlet</servlet-name>
32          <url-pattern>*.xhtml</url-pattern>
33      </servlet-mapping>
34
35  </web-app>
```

Lo único destacable es la línea 13 donde indicamos la variables `PROJECT_STAGE`. Esto nos sirve para indicar si estamos en desarrollo, en producción, ... Esto va ha hacer que determinadas piezas o componentes se comporten de forma distinta. Por ejemplo hay ciertos componentes que en producción darán una excepción ante un error, mientras que en desarrollo sacarán un mensaje en el propio HTML. Esta variable también se puede definir por JNDI, cosa que sería casi más recomendable, para no tener que modificar el fichero cada vez que queremos hacer una instalación en un entorno diferente.

También vemos en la línea 20 la variable `FACELETS_SKIP_COMMENTS`. Esta nos permite indicar que los comentarios de XML `<!-- -->` no se interpreten. En Facelets, por defecto, se interpreta todo, incluso los comentarios. Si queremos que algo no se interprete por el motor de Facelets habría que usar la etiqueta `<ui:remove>`, pero a mi me resulta más cómodo e intuitivo usar los comentarios de verdad, así que prefiero activar este parámetro.

4. Creando nuestra plantilla para la aplicación

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.

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

Una de las novedades es que Facelets ya está integrado dentro del propio estándar (antes lo podíamos usar pero eramos nosotros los que nos teníamos que encargar de la integración ya que era un JAR "de terceros").

Facelets es un sistema de plantillas que nos va a permitir tanto crear plantillas o layouts de nuestra aplicación, como crear nuestros propios componentes por composición de forma muy sencilla (podemos hacer un componente como la suma/composición de otros componentes más pequeños).

Otra de las ventajas de usar Facelets es que nos libramos de las dichosas JSP, y no es que tenga nada en contra de ellas, pero su ciclo de vida no casa con el del diseño en base a componentes de JSF. Así, a partir de ahora, nos vamos a dedicar a escribir XHTML (es decir ficheros XML bien formados y válidos). Además Facelets procesan estos ficheros más rápido que las clásicas JSP, por lo que todo son ventajas ;)

4.1. defaultLayout.xhtml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd ">
03
04 <html xmlns="http://www.w3.org/1999/xhtml " xmlns:ui="http://java.sun.com/jsf/facelets "
05     xmlns:f="http://java.sun.com/jsf/core " xmlns:h="http://java.sun.com/jsf/html ">
06
07 <h:head>
08     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
09     <h:outputStylesheet library="css" name="style.css" />
10     <title>Autentia JSF 2 Tutorial</title>
11 </h:head>
12
13 <h:body>
14 <div id="layout">
15
16     <div id="top" class="top">
17         <ui:insert name="top">
18             <span id="title">Autentia JSF 2 Tutorial</span>
19             <a href="http://www.adictosaltrabajo.com " target="_blank">
20                 <h:graphicImage library="img" name="logoAdictosAlTrabajo.png" alt="Adictos Al
   Trabajo" height="31" />
21             </a>
22         </ui:insert>
23     </div>
24
25     <div id="content">
26         <h1><ui:insert name="title">??? View title ???</ui:insert></h1>
27
28         <ui:insert name="content">??? View content ???</ui:insert></div>
29
30         <div id="bottom">
31             <ui:insert name="bottom">
32                 <span id="copyright">Esta aplicación se distribuye bajo licencia < a
   href="http://www.gnu.org/licenses/gpl-3.0-standalone.html ">GPL</a></span>
33                 <a href="http://www.autentia.com" target="_blank">
34                     <h:graphicImage value="#{resource['img:poweredByAutentia_100x30x72dpi.png']}"
   />
35                 </a>
36             </ui:insert>
37         </div>
38     </div>
39 </h:body>
40 </html>
```

4.2. Gestión de recursos

No vamos a explicar como funciona el sistema de plantillas de Facelets, porque eso no es nuevo, pero si vamos a destacar algunos elementos:

Línea 09 `h:outputStylesheet` - Nos permite indicar que CSS queremos añadir en nuestra página. Por defecto esta CSS se añadirá en el HEAD del html pero con un atributo podemos forzar que se incluya en el BODY. Aunque hemos puesto la etiqueta dentro de `h:head`, esto no quiere decir nada, la propia etiqueta `h:outputStylesheet` determina donde se acabará pintando en el html final la inclusión de la CSS. Las CSS por defecto se incluyen en el HEAD, pero tenemos otra etiqueta para incluir JavaScript (`h:outputScript`) que por defecto lo incluyen en el BODY.

4.2.1. ¿Qué significa el atributo library y name?

JSF 2 va a servir los recursos de un directorio `resources` que puede estar situado en nuestro directorio webapp (donde tenemos las imágenes, html, ...) o dentro del directorio `META-INF` (esto da pie a empaquetar recursos dentro de un JAR).

El atributo `library` indica un subdirectorio dentro de este directorio `resources`, y el atributo `name`, el nombre del recurso que se quiere servir. De esta forma `library="css" name="style.css"` buscará el fichero `resources/css/style.css`

Línea 20 `h:graphicImage` - Es la misma idea que antes, pero para mostrar imágenes. Vemos como sigue la misma idea de `library` y `name`.

Línea 34 `#{resource['img:poweredByAutentia_100x30x72dpi.png']}` - es otra forma de hacer referencia a

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

2010-01-07
Importar el correo de Microsoft Outlook al cliente de correo de Mac OS.

2010-01-07
Monitor de Hudson para Eclipse.

2010-01-07
Patrones de diseño de XML Schema

2010-01-04
Procesador Inteligente de Eventos (IEP) con OpenESB

2010-01-04
PHP Vs Java

2009-12-29
Tutorial de BPEL con OpenESB (II)

2009-12-29
Tutorial de BPEL con OpenESB (I)

2009-12-28
Pruebas funcionales de servicios web con soapUI

2009-12-28
SoapUI: jugando con web services

un recurso, esta vez usando EL. La idea es la misma, pondremos library:nombreDelRecuros

5. Creando la página de login

5.1. loginView.xhtml


```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd ">
03
04 <html xmlns="http://www.w3.org/1999/xhtml " xmlns:ui="http://java.sun.com/jsf/facelets "
05     xmlns:f="http://java.sun.com/jsf/core " xmlns:h="http://java.sun.com/jsf/html ">
06
07 <ui:composition template="./defaultLayout.xhtml">
08     <ui:define name="title">Login</ui:define>
09
10     <ui:define name="content">
11         <h:form>
12             <h:panelGrid columns="3">
13                 <h:outputLabel for="name" value="Nombre" />
14                 <h:inputText id="name" label="Nombre" value="#{loginView.name}">
15                     <f:validateBean />
16                     <f:ajax render="nameError" />
17                 </h:inputText>
18                 <h:message id="nameError" for="name" errorClass="error" />
19
20                 <h:outputLabel for="password" value="Clave" />
21                 <h:inputSecret id="password" label="Clave" value="#{loginView.password}">
22                     <f:validateBean />
23                     <f:ajax render="passwordError" />
24                 </h:inputSecret>
25                 <h:message id="passwordError" for="password" errorClass="error" />
26             </h:panelGrid>
27
28             <h:commandButton action="listTutorialsView" value="Entrar" />
29         </h:form>
30     </ui:define>
31 </ui:composition>
32 </html>
```

5.2. Validaciones según la JSR-303

En la línea 15 nos encontramos con **`<f:validateBean />`**. Con esta nueva etiqueta estamos indicando que queremos usar una validación de Bean, es decir, que queremos usar una validación basada en la JSR-303.

Estas validaciones se definen con anotaciones en el propio Bean, por lo que habría que mirar el código de la clase `LoginView.java` para encontrarnos con:

```
1 ...
2 @NotNull
3 @Size(min = 6, max = 128)
4 private String name;
5
6 @NotNull
7 @Size(min = 6, max = 128)
8 private String password;
9 ...
```

Se ve como estamos validando los atributos para que no sean nulos y para que tengan un tamaño entre 6 y 128 caracteres.

Ojo, porque existe una limitación y es que aunque indiquemos el `@NotNull`, si queremos que JSF 2 nos haga esta

validación, tendremos que indicar en el componente el atributo `required="true"` como hacíamos con JSF 1.

5.3. Soporte para AJAX

JSF 2 ya viene con soporte para AJAX (este está basado en el soporte proporcionado por RichFaces). De esta forma tenemos una nueva etiqueta **`f:ajax`** que pondremos en el componente donde queremos tener comportamiento AJAX; de forma que si la ponemos en un `h:commandButton` se disparará al pulsar el botón, y si la ponemos en un `h:inputText` se disparará al cambiar de valor (cuando el input text pierde el foco).

Básicamente lo que vamos ha hacer con esta etiqueta es indicar que otros componentes queremos que se repinten cuando se produzca el evento (al pulsar el botón, al cambiar de valor, ...). Esto lo haremos con el atributo `render`.

En el ejemplo, en las líneas 16 y 23 se ve como estamos indicando que queremos repintar el componente que presenta los mensajes de error de la validación. Como el `f:ajax` lo hemos asociado a un `h:inputText`, lo que va a ocurrir es que cuando este campo de entrada pierda el foco, se lanzará un evento de cambio de valor, se irá al servidor, se hará la validación de bean según el JSR-303, y si se producen errores estos se pintarán en el componente `h:message` indicado (nótese que con el atributo `render` de `f:ajax` se indican los ids de los componentes que se quieren repintar).

5.4. Navegación

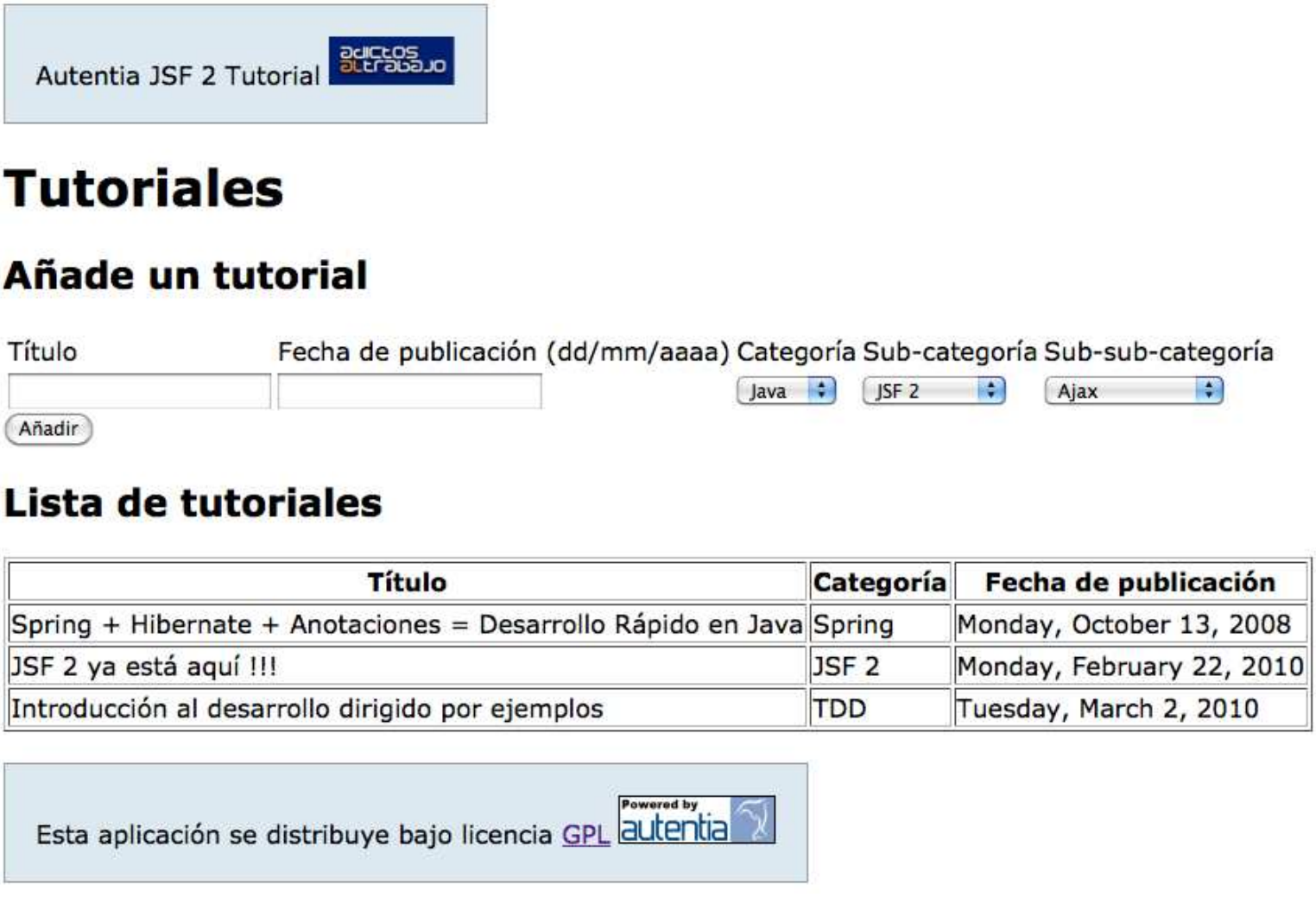
Con JSF 2 se simplifica enormemente la navegación. Podemos seguir usando las reglas de navegación en el `faces-config.xml`, pero JSF 2 añade soporte para "Convención frente a Configuración".

En la línea 28 vemos como en el `h:commandButton`, en el atributo `action`, indicamos una cadena. Esta no es EL, por lo que no estamos haciendo referencia a un backbean. Esta cadena correspondería con el "outcome" que serviría para determinar la regla de navegación a disparar. Pero como no hemos escrito ninguna regla de navegación ¿qué es lo que va ha hacer JSF 2? Sencillo, simplemente se limitará a buscar una página con el mismo nombre y la extensión `.xhtml`. Es decir, si en nuestro ejemplo hemos puesto `action="listTutorialsView"`, JSF 2 intentará saltar a la vista `listTutorialsView.xhtml`

En el ejemplo la página se buscará a la misma altura, pero podemos indicar rutas relativas. Por ejemplo, si la ruta empieza por / estaremos indicando que es relativa al dominio y nombre de aplicación actual.

6. Creando la página de alta y listado

6.1. listTutorialsView.xhtml



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd ">
```

```
03
04 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:ui="http://java.sun.com/jsf/facelets"
05     xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html">
06
07 <ui:composition template="./defaultLayout.xhtml">
08     <ui:define name="title">Tutoriales</ui:define>
09
10     <ui:define name="content">
11         <h2>Añade un tutorial</h2>
12         <h:form>
13             <h:panelGrid columns="5">
14                 <h:outputLabel for="title" value="Título" />
15                 <h:outputLabel for="publicationDate" value="Fecha de publicación (dd/mm/aaaa)" />
16                 <h:outputLabel for="category" value="Categoría" />
17                 <h:outputLabel for="subCategory" value="Sub-categoría" />
18                 <h:outputLabel for="subsubCategory" value="Sub-sub-categoría" />
19
20                 <h:inputText id="title" label="Título" value="#{listTutorialsView.title}"
   required="true" />
21                 <h:inputText id="publicationDate" label="Fecha de publicación"
   value="#{listTutorialsView.publicationDate}" required="true">
22                     <f:convertDateTime pattern="dd/MM/yyyy" />
23                 </h:inputText>
24                 <h:selectOneMenu id="category" label="Categoría"
   value="#{listTutorialsView.category}" converter="selectItemsConverter">
25                     <f:selectItems value="#{listTutorialsView.categories}" var="category"
   itemLabel="#{category.name}" />
26                     <f:ajax render="subCategory subsubCategory" />
27                 </h:selectOneMenu>
28                 <h:selectOneMenu id="subCategory" label="Sub-categoría"
   value="#{listTutorialsView.subCategory}" converter="selectItemsConverter">
29                     <f:selectItems value="#{listTutorialsView.subCategories}" var="subCategory"
   itemLabel="#{subCategory.name}" />
30                     <f:ajax render="subsubCategory" />
31                 </h:selectOneMenu>
32                 <h:selectOneMenu id="subsubCategory" label="Sub-sub-categoría"
   value="#{listTutorialsView.subsubCategory}" converter="selectItemsConverter">
33                     <f:selectItems value="#{listTutorialsView.subsubCategories}"
   var="subsubCategory" itemLabel="#{subsubCategory.name}" />
34                 </h:selectOneMenu>
35             </h:panelGrid>
36             <h:commandButton action="#{listTutorialsView.add}" value="Añadir" />
37             <h:messages errorClass="error" />
38         </h:form>
39
40         <h2>Lista de tutoriales</h2>
41         <h:dataTable id="tutorials" value="#{listTutorialsView.tutorials}" var="tutorial"
   border="1">
42             <h:column>
43                 <f:facet name="header">
44                     <h:outputText value="Título" />
45                 </f:facet>
46                 <h:outputText value="#{tutorial.title}" />
47             </h:column>
48             <h:column>
49                 <f:facet name="header">
50                     <h:outputText value="Categoría" />
51                 </f:facet>
52                 <h:outputText value="#{tutorial.category.name}">
53                 </h:outputText>
54             </h:column>
55             <h:column>
56                 <f:facet name="header">
57                     <h:outputText value="Fecha de publicación" />
58                 </f:facet>
59                 <h:outputText value="#{tutorial.publicationDate}">
60                     <f:convertDateTime dateStyle="full" />
61                 </h:outputText>
62             </h:column>
63         </h:dataTable>
64     </ui:define>
65 </ui:composition>
66 </html>
```

6.2. Haciendo nuestros propios conversores

En la línea 24 vemos como indicamos un conversor para el `h:selectOneMenu` con el atributo `converter="selectItemsConverter"`. Pero ¿cómo hemos declarado y registrado este conversor? Para contestar a esta pregunta tenemos que ver el código de la clase `SelectItemsConverter.java`, donde nos encontraremos con:

```
1 ...
2 @FacesConverter("selectItemsConverter")
3 public class SelectItemsConverter implements Converter {
4 ...
```

De nuevo JSF 2 nos simplifica la vida, ya que gracias a la anotación `@FacesConverter` no tenemos que dar de alta el conversor en el fichero de configuración `faces-config.xml`. El valor de la anotación será el id del conversor, por el cual haremos referencia en el xhtml (línea 24)

Al igual que esta anotación tenemos otras que nos permiten registrar validadores, componentes, ...

6.3. Listas desplegables relacionadas en cascada

No se si os ha pasado alguna vez, pero es bastante común tener listas desplegables relacionadas entre si. Lo típico de selecciono país, selecciono región, selecciono ciudad; o cosas por el estilo. Dentro de que esta situación es bastante común, no era del todo fácil de implementar, y nos podía dar más de un quebradero de cabeza (que si pongo el immediate="true", que si cambio los valores de las lisas, ...).

Con JSF 2 estos problemas son historia, y ahora gracias al soporte para AJAX se convierte en algo trivial. En las líneas 26 y 30, podéis ver como volvemos a hacer uso de `f:ajax` para indicar a JSF que cuando se produzca un evento de cambio de valor en la lista se vaya al servidor y a la vuelta vuelva a repintar las otras listas. En la línea 26 vemos como indicamos dos ids en el atributo render, esto es perfectamente posible, de forma que si queremos repintar más de un componente, basta con indicar la lista de ids separados por un espacio en blanco.

6.4. Ámbito de los BackBeans

Con JSF teníamos los ámbitos de request (el ámbito por defecto), session y application. En JSF 2 tenemos un par más: view y custom (el ámbito custom nos permite definir nuestros propios ciclos de vida para los backbeans).

View es más que request y menos que session. Es decir la primera vez que naveguemos a una vista (a una página) se creará el backbean, y este se mantendrá hasta que saltemos a otra vista por una regla de navegación. Esto quiere decir que todas las peticiones AJAX, o incluso peticiones que no sean AJAX pero que no nos hagan saltar de vista, compartirán la misma instancia del backbean.

Este comportamiento es especialmente indicado para situaciones como la que hemos descrito en el punto 6.3. Si los valores de las listas desplegables están relacionadas, al cambiar el valor de la primera lista, se cambiará el valor de las otras dos. Si el backbean está en request al hacer la siguietne petición hemos perdido el estado, por lo que era causa de bastantes errores, y lo que se solía hacer era guardar el backbean en session. Gracias al ámbito view, ya no es necesario saturar la session de backbeans para solucionar estos problemas.

Para ver como podemos indicar que el backbean es de ámbito view, podemos ver el código de la clase ListTutorialsView.java:

```
01 ...
02 @ManagedBean
03 @ViewScoped
04 public class ListTutorialsView implements Serializable {
05
06     ...
07
08     @ManagedProperty("#{libraryOfAlexandria}")
09     private LibraryOfAlexandria libraryOfAlexandria;
10     ...
```

En la línea 03 vemos como con la anotación @ViewScoped indicamos que el backbean es de ámbito view. De nuevo no ha hecho falta declarar nada en el faces-confg.xml. Por supuesto tenemos anotaciones similares para el resto de ámbitos.

6.5. Declaración de los backbeans e Inyección de Dependencias

En JSF 1 teníamos que declara los backbeans en el fichero `faces-config.xml`. En JSF 2, gracias a la anotaciones `@ManagedBean`, ya no es necesario. Podemos ver un ejemplo de esto en la línea 02 del código del apartado 6.4. Si no indicamos nada en la anotación, el id que se le dará al backbean será el de la clase poniendo la primera letra en minúscula.

Ademas con anotaciones también podemos hacer Inyección de Dependencias. Por ejemplo, en la línea 08 podemos ver como con `@ManagedProperty` estamos inyectando el resultado del EL "`#{libraryOfAlexandria}`", esto es, un backbean cuyo id es "`libraryOfAlexandria`".

La Inyección de Dependencias de JSF 2 tiene algunos inconvenientes o incomodidades. Por ejemplo, aunque anotamos el atributo de la clase, estamos obligados a tener el método getter y setter.

Y otra cosa más importante es que JSF 2 sólo es capaz de buscar los `@ManagedBean` que están en el directorio `classes` de nuestra aplicación web. Es decir, no nos podríamos inyectar un bean que esté dentro de un JAR. Debido a esta limitación la clase `LibraryOfAlexandria` la hemos metido dentro del proyecto web, pero realmente tendría que estar en el proyecto del modelo, ya que se trata de un servicio de negocio (en otro tutorial ya veremos como solventar esta limitación combinando JSF 2 con Spring).

JSF 2, también admite otras anotaciones como el `@PostConstruct`. Esta anotación se pone en un método y le indica a JSF 2 que debe llamar a ese método después de crear e inyectar los valores al backbean. Esta anotación es especialmente útil para hacer tareas de inicialización del backbean usando las dependencias que nos inyecta JSF 2.

7. ¿Y dónde está el fichero de configuración faces-config.xml?

Eso es de las mejores cosas que tiene JSF 2, que no hace falta !!!

Con esto no quiero decir que no lo uses, simplemente usarlo cuando os aporte valor, pero en este ejemplo hemos visto como podemos hacer una aplicación perfectamente funcional, con soporte para AJAX, con conversores propios, con navegación, ... sin que exista este fichero.

8. Conclusiones

Sólo hemos empezado a rascar y ya nos gusta lo que encontramos.

Se dice que JSF 1 es un Ivory Tower, es decir, que está pensado desde un punto de vista intelectual, pero desconectado del mundo práctico. El papel lo agunta todo, pero cuando lo llevamos a la práctica hay cosas que no resultan tan útiles o tan sencilla. De hecho por eso nos animamos a crear Wuija (http://sourceforge.net/projects/wuijaframework/), para solventar todas estas deficiencias.

Ahora JSF 2 se construye bajo la experiencia práctica de librerías como ICEfaces (http://www.icefaces.org), RichFaces (http://www.jboss.org/richfaces), PrimerFaces (http://www.primefaces.org/), MyFaces (http://myfaces.apache.org/tomahawk/index.html), y casi podríamos decir que es un compendio de buenas ideas de todas ellas. Consiguiendo de esta manera simplificar el desarrollo y afianzar el estándar como una opción de futuro.

En próximos tutoriales, desde Autentia (http://www.autentia.com), intentaremos enseñaros a hacer componentes por composición (nueva característica de JSF 2 que tampoco estaba anteriormente en Facelets), a integrar con Spring 3, a integrar con otras librerías de componentes, y espero que a muchas cosas más ... ;)

9. Sobre el autor

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

Socio fundador de Autentia (Formación, Consultoría, Desarrollo de sistemas transaccionales)

mailto:alejandropg@autentia.com

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

http://www.autentia.com

10. Recursos de interes

- http://www.laliluna.de/articles/jsf-2-evaluation-test.html
- http://andyschwartz.wordpress.com/2009/07/31/whats-new-in-jsf-2/
- http://java.sun.com/javaee/6/docs/tutorial/doc/bnaph.html
- http://java.sun.com/javaee/javaserverfaces/2.0/docs/pdldocs/facelets/index.html
- http://java.sun.com/javaee/javaserverfaces/2.0/docs/api/index.html

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 «

Autor **Mensaje de usuario registrado**

- 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. ADD THIS BLOG TO MY Technorati FAVORITES

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
The JSF 2 Return	En este tutorial veremos algunas de las principales novedades de JSF que viene con más fuerza que nunca	2010-03-09	8	-	-	
Plugin de mejora de graficos para JMeter	Plugin que permite incorporar una nueva vista de los graficos en JMeter	2010-02-17	520	-	-	
Creación de un componente JSF, basádonos en un plugin de jQuery, con el soporte de RichFaces.	Usando un plugin para jQuery que ya conocemos, jcarousel, vamos a generar un componente JSF (facelets) con el soporte de RichFaces	2010-02-04	911	Bueno	1	
Integración de jQuery en RichFaces.	En este tutorial analizamos cómo podemos tener soporte de jQuery dentro del ámbito de componentes JSF de RichFaces.	2010-02-03	760	-	-	
AjaxSingle: el partialSubmit de RichFaces.	En este tutorial vamos a examinar la posibilidad de hacer uso del atributo ajaxSingle, en los componentes de RichFaces, que funciona como el atributo partialSubmit de ICEfaces.	2010-02-02	694	Muy bueno	2	
Introducción a RichFaces.	RichFaces es una librería de componentes visuales para JSF con soporte para Ajax4JSF.	2010-02-01	1140	-	-	
JMeter. Uso de funciones.	En este tutorial tratamos el uso de las funciones más habituales de la herramienta JMeter.	2010-01-26	617	Muy bueno	1	
JMeter. Gestión de usuarios	En este tutorial tratamos la simulación de distintos usuarios, en la herramienta JMeter, mediante el archivo externo users.xml o mediante la función Counter.	2010-01-14	1266	-	-	
JMeter y JSF. Extracción del parámetro ViewState	En este tutorial ofrecemos una solución a la parametrización del atributo ViewState, de JSF (Java Server Faces), cuando ejecutamos scripts de pruebas de carga mediante la herramienta JMeter.	2010-01-11	1120	-	-	
Migración de EJB3 a JPA y Spring.	Este tutorial de como emigrar una aplicación montada con EJB3 y JSF al soporte que proporciona Hibernate para JPA y a Spring, con el mismo soporte de JSF.	2009-07-24	223	Muy bueno	1	

Nota: