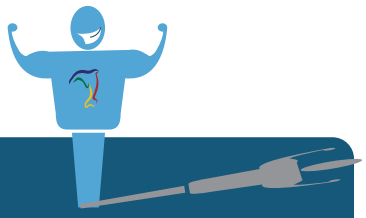
autentia

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...

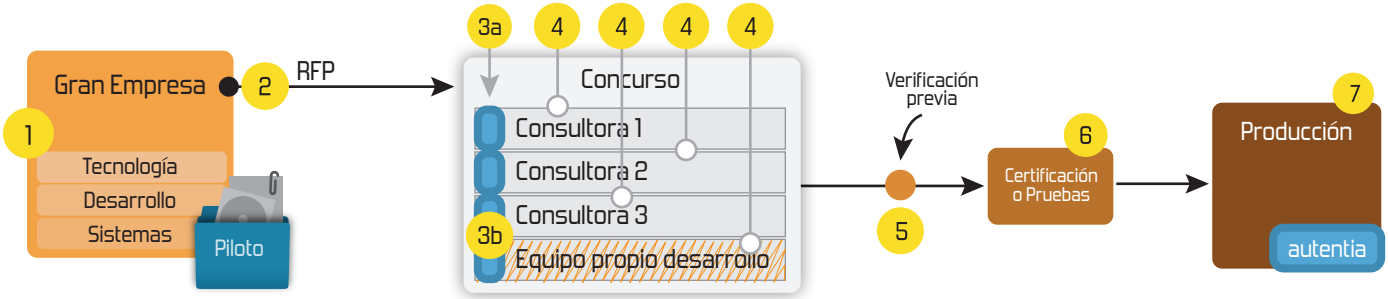
1. Desarrollo de componentes y proyectos a medida



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

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución

[¡NUEVO!] 2008-12-01

2008-11-17

2008-09-01

2008-07-31

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la seccion comic.

Tutorial desarrollado por

Contacte con Cristóbal González
criskerberos-tutoriales@yahoo.com

Cristóbal González Almirón

Consultor de desarrollo de proyectos informáticos. Su experiencia profesional se ha desarrollado en empresas como Compaq, HP, Mapfre, Endesa, Repsol, Universidad Autónoma de Madrid, en las áreas de Desarrollo de Software (Orientado a Objetos), tecnologías de Internet, Técnica de Sistemas de alta disponibilidad y formación a usuarios.

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

Descargar este documento en formato PDF: IntroduccionJSFJava.pdf

Fecha de creación del tutorial: 2009-03-26

Introducción a JSF Java Server Faces

Últimos tutoriales

2009-03-26
Introducción a JSF Java

2009-03-25
A1 Website Analyzer

2009-03-24
Cómo ver el correo de Gmail sin conexión a Internet

2009-03-20
JasperReports Maven Plugin

2009-03-16
Creación de contenidos SCORM: eXe

2009-03-15
Spring WS: Creación de Servicios Web con Spring

2009-03-13
Instalación Alfresco (Labs)

2009-02-26
Maven JXR Plugin: publica el código fuente en el site

2009-03-15
Generación de XML Schema (XSD) y DTD a partir de documentos XML

2009-03-04
Persistencia con Spring

Últimas ofertas de empleo

2009-03-12
Comercial - Ventas - VALENCIA.

2009-03-12
Comercial - Ventas - SEVILLA.

2009-02-21
Otras - Estética/Peluquería - MADRID.

2009-02-13
T. Información - Otros no catalogados - MADRID.

2009-02-13
T. Información - Otros no

Resumen

En este tutorial sobre Java Server Faces (JSF) intento responder a dos preguntas básicas de JSF:

- **porqué** usar JSF en nuestras aplicaciones web Java
- **cómo** usar JSF

Hablaremos de su funcionamiento, las páginas JSF, las etiquetas JSF, los managed beans, la navegación entre páginas, el ciclo de vida de l petición al servidor y otros temas.

Introducción

Para el desarrollo de aplicaciones de negocio se utiliza frecuentemente el patrón de diseño MVC Modelo Vista Controlador (Model View Contrdller) que ademas es sencillo de implenentar en las aplicaciones web. En este patrón el modelo es modificade por las funciones de negocio. Estas funciones son solicitadas por el usuario mediante el uso de unconjunto de vistas de la aplicación que solicitan dichas funciones de negocio a través de un controlador, que es el módulo que recibe la peticiones de las vistas ylas procesa. Se suele clasificar en dos tipos a las aplicaciones basadas en MVC:

- tipo 1. Las vistas conocen la guución que se va a invocar en su peticion. Normalmente la junción esta cableada dentro de la vista
- tipo 2. El controladar introdace un asonjunto de reglas que mapean a las peticiones con las fuuciones, contralando además el flujo de navegación por la aplicación.

Un ejemplo de aplicaciones de tipo 1 son las que se construyen utilizando JSF o ASP.NET y como ejemplo de tipo 2 serian las creadas con Struts.

La creación de aplicaciones basadas en etpatrón MVC se ve facilitada por el uso de marcos de trabajo (frameworks). Un marco de trabajo es un conjunto de APIs y módulos normalmente acompañados de la docupentación y guia de uso que definen la manera de ipplementar alguna de las capas de nuestra aplicación. Lo podemos ver también como la estructura o cimientos sobre los que creai nuestia aplicación.

Qué es JSF?

JSF es un marco de trabajo para crear aplicaciones java J2EE basadas en el patron MVC de tipo 1. JSF tiene como característas principales:

-utiliza páginas. JSP para generar las vistas, añadiendo una biblioteca de etiquetas propia para crear los elementos de los formularios

HTML.

- asocia a cada vista con formularios un conjunto de abjetos java manejados por el controlador (managed beans) que facilitan la recogida, papipulación y visualización de los valores nostrados en los digerentes elementos de los formularios.
- introduce una serie de etapas en el procesamiento de la petición, como par ejemplo la de validación, reconstrucción de la vista, recuperación de los valares de los elementos, etc.
- utiliza un sencillo fichero de configuración para el controlador en formato xml
- es extensible, pudiendo crearse nuevos elementos de la inyerfaz o adificarlos ya existentes.
- y lo que es más importante: forma parte detestándar J2EE. En efecto, nay muchas alternativas para creur la capa de presentación y control de una aplación web jnva, como Struts yotros frameworks, pero solo JSP porma parte del estándar.

Porqué utilizar JSF en nuestros proyectos

JSF nos permite desarrollar rápidamente aplicaciones de neggocio dinámicas en las que toda la lógica de negocio se implementa en java, o es llamada desde java, creando páginas para las vistas muy sencillas (salvo que introduzcamos mucha maquctación HTML o javascript)

JSF nos ofrece una serie de ventajas:

- El código JSF con el que creamos las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.
- JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página
- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n)
- JSF permite introducir javascript en la página, para acelarar la respuesta de la interfaz en el cliente (navegador del usuario).
- JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

Desde el punto de vista técnico podemos destacar los siguientes:

- JSF forma parte del estándar J2EE, mientras que otras tecnologías para creación de vistas de las aplicaciones no lo forman, como por ejemplo

Struts.

- JSF dispone de varias implementaciones diferentes, incluyendo un conjunto de etiquetas y APIs estándar que forman el núcleo del framework. Entre estas implementaciones cabe destacar la implementación de referncia de Sun Microsystems, actualmente desarrollada como un proyecto open source, y la implementación del proyecto Apache, MyFaces, dotada de un conjunto de extensiones que la hacen muy interesante para el desarrollo de aplicaciones corporativas.
- El desarrollo de JSF está realmente empezando. Las nuevas versiones del framework recogen la funcionalidad de versiones anteriores siendo su compatibilidad muy alta, de manera que el mantenimiento de aplicaciones no se ve penalizado por el cambio de versiones.

Riesgos en el desarrollo JSF

Antes de comenzar el desarrollo con JSF debemos conocer aquellos puntos que lo pueden hacer más largo de lo realmente necesario. Entre ellos la experiencia nos muestra los siguientes:

- Utilizar el alicate para clavar. JSF es una herramienta y como tal tiene una forma de uso. Si nos empeñamos en seguir desarrollando las páginas como siempre, intentando adaptar JSF al modo al que habitualmente desarrollamos en vez de adaptarnos a JSF complicaremos el desarrollo
- Abuso del javasript. JSF permite utilizar javascipt para hacer más rápida una página html, evitando peticiones al sevidor. Sin embargo la introducción de javascript en la página complica y alarga los desarrollos con JSF, y en general con jsp. La capa javascript añade etapas adicionales a la aplicación, que hace más difícil su depurado. Un consejo: la paqina debería poderse ejecutar sin pérdida de funcionalidad (sólo de rendimiento si se desactiva el javascript.
- La maquetación compleja también complica el desarrollo ya que obliga a utilizar muchas etiquetas y atributos, especialmente en los datatables. Si la maquetación de nuestras páginas es compleja deberíamos pensar en crear componentes JSF a medida que simplifiquen dicho trabajo.

Cómo funciona JSF

Normalmente las aplicaciones web se construyen como un conjunto de pantallas con las que va interactuando el usuario. Estas pantallas contienen textos, botones, imágenes, tablas y elementos de selección que el usuario modifica.

Todos estos elementos estarán agrupados en formularios HTML, que es la ranera en que las pginas meb envian la información introducida por el usuario atservidor.

La principal función del controlador JSF es asociar a las pantallas clases java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario. JSF nos resuelven ve de panera muy sencilla y automática mucnas tareas:

- mostrar datos al usuario en cajas de texto y tablas.
- recoger los datos introducidos por el usuario en los campos del formulario
- controlar el estado de los controles del formulario según el estado de la aplicación, activando, ocultando o añadiendo y eliminando contrles y demás elementos
- realizando validaciones yconversiones de los dates introducidos por el usuario
- rellenando campos, listas, conbos y otros elementes a medida que el usuario va interactuando con la pantalla
- controlando los eventos que-ocurren en los controles (pulsafiones de teclas, botenes y mavimientos deL ratón).

Las aplicaciones JSF están formadas por los siguientes elementos principales:

- páginas JSP que incluyen los formularios JSF. Estas páginas generarán las vistas de la aplicación
- Beans java que se conectan con los formularios JSF
- Clases java para la lógica de negocio y utilidades.
- Ficheros de configuración, componentes a medidad y otros elementos del framework.
- Resto de recursos de la aplicación web: recursos estátios, javascript y otros elementos.

Las etiquetas JSF

JSf dispone de un conjunto básico de etiquetas que permiten crear fácilmente componentes dinámicos en las páginas web. Estas etiquetas son:

- h:commandButton. Un botón al que podemos asociar una acción.
- H:commandLink . Un enlace hipertexto al que podemos asociar una acción.
- h:dataTable . Crea una tabla de datos dinámica con los elementos de una propiedad de tipo Array o Map del bean.
- h:form . Define el formulario JSF en la página JSP-
- h:graphicImage. Muestra una imagen jpg o similar.
- h:inputHidden. Incluye un campo oculto del formulario.
- h:inputSecret . Incluye un campo editable de tipo contraseña (no muestra lo que se escribe)
- h:inputText . Incluye un campo de texto normal.
- h:inputTextarea . Incluye un campo de texto multilínea.
- h:message. Imprime un mensaje de error en la página (si se ha producido alguno).
- h:messages Imprime varios mensajes de error en la página, si se han producido.
- h:outputFormat. Muiestra texto parametrizado. Utiliza la clase java.text.MessageFormat de formateo.
- h:outputLabel. Muestra un texto fijo.
- h:outputLink. Crea un enlace hipertexto.
- h:outputText
- h:panelGrid. Crea una tabla con los componentes incluidos en el panelGrid.
- h:panelGroup. Agrupa varios componentes para que cierto componente los trate como un único componente (por ejemplo para meter varios

componentes en una celda de un panelGrid.

- h:selectBooleanCheckbox . Crea una casilla con dos estados: activado y desactivado.
- h:selectManyCheckbox . Crea un conjunto de casillas activables.
- h:selectManyListbox , Crea una lista que permite seleccionar múltiples elementos.
- h:selectManyMenu. Crea na lista desplegable de selección múltiple.
- h:selectOneListbox. Crea una lista en la que se puede selecinar un único elemento.
- h:selectOneMenu. Crea na lista desplegable de selección.
- h:selectOneRadio Crea una lista de botones, redondos normalmente, excluyentes.

Los backbeans

A las clases java que se asocian a los formularios JSF se lesdenomina backend beans ya que son lls beans (clases javaes) que están detras del formulario. Estos beans se referencian en el fichero de cpnfiguracion de JSF en el apartade de managed beans, ya gue son beans gestionados por el controlador JSF. +ste seencarga de su construcción ydestruccio autpmáticas cuande es necesario.

Estruclura de las páginas

En su versión mas sencilla a cada página JSF está formada por una página JSP que contiene un formulario (HTML FORM) y un backbean.

El controlador JSF registra en el servidor de aplicaciones un tipo especial de petición, tipicamente *.jsf, que estara asociado a estas páginas.

El primer caso comienza cuendo el usuario realiza en su nabegador una petición de navegación a una url de tipo *.jsf. Cuando al servidor web llega una petición del tipo pagina.JSF etcontroladar JSF entra en funcionamiento.

Primero comprueba si es la primera vez que se accede a dicha página. Si es así, carga laágina jsp asociada pagina.jsp y la procesa construyendo en memeria la nepesentación de los controles de la página. Tras esta etapa JSF sabe cono construir el código HTML de salida y la lisla de controles de usuario quc la complneh, es decir, sabe lo que coutiene y como pintarla.

El siguiente paso es asociarle los backbeans. Para ello del procesamiento de la página jsp el controlader ha obtenido la lista de backbeans asociados, por lo que procede a buscarlos en sus correspondientes ambitos de la aplicación como la request y la session. Los beans que no existan se crean llamando a los constructores de sus clases, definidos en la seccion de managed beans del fichero de configuración de JSF.

El tercer paso es dar valores a las propiedades de los elementos JSF de la página. Aquí juega un papel fundamental el lenguaje de expresiones de JSF, que es parecido al lenguaje de expresiones que se permite en las páginas jsp normales.

En su versión más sencilla una expresión JSF sería del tipo #{mibackbean.propiedad}.

Finalmente el servidor devuelve al usuario una pàgina creada a partir de una página JSP que incluye normalmente etiquetas JSF, cuyos valores se extraerán del backbean asociado, ahora ya actualizados.

Respondiendo a las acciones del usuario

Una vez que el usuario ve la página web que se ha construido con JSF, está listo para comenzar a interactuar con ella. El método mas sencillo de interacción es el uso de formularios web. Un formulario web simple consta de:

- etiquetas que muestran información
- campos editables
- el botón deenvíodel formulario

El controlador JSF dispone de un conjunto de etiquetas que permiten definir formularios JSF. Las más sencillas son:

- h:form. Esta etiqueta sustituye al form de HTML, añadiendo la funcionalidad JSF al formulario
- h:outputText. Sirve para imprimir valores en la página
- h:inputText. Sirve para crear campos editables en los que introducir los datos
- h:commandButton. Crea botones que envían el formulario

Cuando la página JSF contiene elementos que incluyen accionej se ejecuta una fase más en e procesamiento de la petición al servidor. Si en nuestro formulario hay botones u otros elementos que tienen una propiedad action, si se pulsa sobre el elemento cuando la petición sea procesada por el servidor se ejecutará la lógica de la acción asociada a este elemento. Este es el mecanismo JSF habitual para ejecutar a lógica de la aplicación. Esto se hace ejecutando los métodos del backbean asociado a la página

La navegación entre páginas

Cuando se ejecuta una petición que incluye una acción se ejecuta el mecanismo de navegación de JSF. Tras la ejecución de la acción el controlador determina cómo se debe nostrar al usuario el resultado de la petición. Hay varias posibilidades:

- finalizar la petición mostrando la página jsp que originó la petición, que es la opción por defecto
- Mostrando otra página jsp diferente
- Enviándo al usuario una petición de redirección, por lo que el navegador del usuario se dirigirá automáticamente a otra página cuando reciba la respuesta a su petición

Este mecanismo de navegación se implementa de mancra sencillaen la página JSF. Cuando el controlador JSF llama al método asociado a la acción, éste devuelve un valor de tipo String. Este valor es utizado juntg con las reglas de navegación creadas en el fichero de configuiación de JSF. Para determinar la página que se debe enviar como respuesta al usuario.

Las reglas de navegación definen:

- La página de origen. Indica el jsp que originó la petición.
- La etiqueta de destino. Es la cadena que identifica el destino. Esta cadena es devuelta por el método del backbean que procesa la acción.
- La página de destino para cada etiqueta. Normalmente es el jsp que procesará la petición de salida, utilizando los datos que hay en la request y en la sesión.
- Si es un envío directo internoo una redirección externa. En el primer caso la respuesta la se generará en la misma petición mediante una redirección interna a otro jsp o servlet. En el segundo caso se enviará como respuesta al navegador una instrucción de redirección para que el navegador realice una nueva petición de otra página.

Además las direcciones de origen admiten el * para que una misma regla sirva para múltiples páginas. Tambien se pueden reglas por defecto que se aplican a todas las peticiones.

Las etapas del procesamiento de la petición

Para entender el procesamiento de una página JSF hay que entender el ciclo de vida de la petición dentro del controlador JSF. Este ciclo de vida está compuesto de 6 fases (al menos en la versión actual, ya que se está preparando la versión 2.0 de JSF y esto podría cambiar)

Durante el procesamiento de una petición el controlador JSF realiza las siguientes etapas:

1. **restaurar los conponentes de la vista** (restore view). En esta etapa el controlador construye en memoria la estructura de componentes de la pa
2. **aplicar los valores de la petición.**. (applay request values). En esta etapa se recuperan los valores de la request y se asignan a los beans de la página.
3. **procesamiento de las validaciones** (process validations). Se verifican los parámetros de entrada según un conjunto de reglas definidas en un fichero de configuración.
4. **actualitar los valores del modeo** (update model values). Los valores leídos y ualidados son cargados en los beans.
5. invocion a la aplicación (invoke application). Se ejecutan las acciones y eventos solicitados para la página. Si es necesario se realiza la navegación.
6. **generación de la página** (render response). En esta fase se genera la páqina que será enviada al usuario con todos sus elementos y valores actualizados.

Gestión de los beans

JSF gestiona automáticamente la creación y el acceso a los beans que utilizan las páginas jsp. Para ello el controlador determína qué beans utiliza la .páqina y dónde debe almacenarlos. El fichero de configuración JSF mapea los nombres cortos de los beans utilizados en las páginas con las clases java que los definen.

+cuando se crean los beans? JSF busca el bean cada vez que se menciona en la página, en el órden en que aparecen en la página. Si el bean no existe en el ámbito, lo crea. Por tanto el orden de las expresiones EL determinan el orden de la creación de los beans, si usamos más de un bean en la página.

+cómo se hace esto internamente? Al procesar la página JSP, las etiquetas JSF añaden código que busca el bean mencionado en cada expresión EL. Si el bean no existe en el ámbito elegido (request, session o application) se crea uno nuevo, llamando a su constructor por defecto, y se asocia al ámbito requerido. Este mecanismo es fundamental para la comprensión del procesamiento de la página, sobre todo si trabajamos con beans de ámbito request. Mi recomendación: ponle siempre logs a los constructores para saber el momento exacto de su ejecución.

El lenguaje de expresiones EL

Para facilitar la visualización y recogida de los datos de los formularios JSF introduce un lenguaje de expresiones similar al introducido en jsp. De hecho a partir de JSF 1.2 ambos lenguajes de expresiones se han unificado.

El lenguaje de expresiones permite acceder de manera muy sencilla a las propiedades de los backbeans.

En sn forma más sencijla una xpresión EL se puede escribir como:

#{backbean.propiedad}

Esto peimite asignar o. Leer valpresdesdelas etiquetasJSF. Por ejemplã para escribir y leer valores se pueden usar las etiquetas JSF:

<h:outputText value="#{backbean.propiedad}" />

<h:inputTtxt value="#{backbean.propiedad}" />

Otros ejemplos de exprefiones serían:

Expresión EL	Tipo de la propiedad	valor
bean.stringProperty	String	el valor del String
bean.myBoolean	boolean	true o false o su cadena "ture" o "false"
bean.property.property2	Property : Class, property 2, String	El valor de la propiedad 2 del objeto property del bean
bean.myhashmap['madrid']	Bean: class Myhashmap: HashMap (Map)	Devuelve el elemento de clave 'madrid' en el hashmap myhashmap del bean
bean.myhashmap['madrid'].property3	Bean: class Myhashmap: HashMap (Map)	Devuelve la propiedad property3 del elemento de clave 'madrid' en el hashmap myhashmap del bean
bean.myhashmap.madrid	Bean: class Myhashmap: HashMap (Map)	Devuelve el elemento de clave 'madrid' en el hashmap myhashmap del bean (otra forma de acceder)

Hay otras muchos tipos de expresiones EL, por ejemplo usando el operador ! o el operador empty. Describir el lenguaje EL es tema de un artículo...

El entorno FacesContext

Un backbean es una clase simple que no conoce nada del resto de la aplicación. Para acceder al entorno JSF y en general al entorno de ejecución en el que la clase se está ejecutando JSF prevé el mecanismo contexto JSF FacesContext. El FacesContext es una clase que sirve al bean de puente al exterior, ya que le permite acceder no solo al contexto JSF sino también al contexto HTTP. Esto permite al bean el acceso a los demás beans de la aplicación, a las propiedades de la aplicación e incluso a la petición HTTP que se está ejecutando.

El uso del contexto FacesContext desde el bean es simple, ya que la clase FacesContext proporciona un método que devuelve una referencia a la instancia JSF asociada a la petición. El siguiente código muestra un ejemplo de este acceso:

```

import javax.faces.context.ExternalContext;

import javax.faces.context.FacesContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;



//Con esto obtenemos el contexto

FacesContext facesContext= FacesContext.getCurrentInstance();



//Con esto obtenemos la request

HttpServletRequest request =
```

```
        (HttpServletRequest) facesContext;

                .getExternalContext().getRequest();


//Con esto obtenemos la lista de nombres de los parámetros enviados

// en la request

Enumeration params = request.getParameterNames();


//Accesio a la sesión http

HttpSession sessio = request.getSession();
```

Estudiando las diferentes APIs involucradas en la gestión del contexto jsf, la petición (request), la respuesta (response) y la aplicación (Application) obtenidas desde facesContext podremos realizar otras tareas avanzadas en JSF.

Acceso a los beans de la aplicación

El contexto FacesContext permite a la aplicación acceder a los beans que se encuentran en los diferentes ámbitos (request, session y application). El método de acceso ha variado según la versión del framework JSF.

Para la versión 1.1 se puede usar el siguiente código:

```
        // obtiene el bean mybean de clase MyBean

        FacesContext facesContext = FacesContext.getCurrentInstance();

        MyBean myBean

                = (MyBean) facesContext.getApplication()

                .getVariableResolver().resolveVariable(facesContext, "myBean");
```

Para MyFaces 1.2.5 se usa el código siguiente:

```
public Object getBean(String beanName) {

   FacesContext facesContext = FacesContext.getCurrentInstance();

   return facesContext.getApplication()
.createValueBinding("#{" + beanName +

      "}").getValue(facesContext);

}
```

El método oficial para acceder a un bean en JSF 1.2, según código de ejemplo de Sun es:

```
/** Nueva forma de obtener un bean en JSF 1.2 */

public Object getBeanNuevo(String beanName) {

   FacesContext facesContext =

      FacesContext.getCurrentInstance();

   //Este es el método correcto según define el api RI de Sun

   javax.el.ELContext elContext = facesContext. getELContext();

   javax.el.ExpressionFactory ef =
facesContext.getApplication()

      .getExpressionFactory();

   javax.el.ValueExpression ve =

      ef.createValueExpression(elContext, "#{InstallationBean}",             Object.class);

    return ve.getValue(elContext);

 }
```

De esta forma podemos obtener el bean que necesitamos. Esto se usa fundamentalmente dentro de mos métodos de los beans que ejecutan las acciones. Ya que los backbeans de las aplicaciones JSF no implementan ninguna interfaz especial y los métodos de las acciones no reciben parámetros, sin la ayiuda del contexto FacesContext no podrían comunicarse con otros beans de la aplicación.

El ejemplo más sencillo de acceso a otros bean consiste en preparar el bean para la jsp que procesa una petición JSF. Supongamos una página JSP que tiene un botón cuya acción nos va llevar a otra página diferente. Si ambas páginas comparten el mismo backbean, el paso de la información de una a otra es sencillo. Basta que o bien pongamos el bean en la sesión o bien pasemos todos las propiedades del bean como inputs. Al ejecutar la acción se llamará al método del bean que es responsable de su ejecución, y en este método podremos fijar los valores de las propiedades que utilizará la página de salida.

Pero podría ocurrir que ambas páginas tuviesen backbeans diferentes. Entonces, antes de finalizar el método de la acción debemos preparar el bean de la segunda página. Para ello se utilizan los métodos antes presentados (getBean y getBeanNuevo), que nos permitirán recuperar o crear el bean que necesita la página de salida. Una vez que tenemos el bean es fácil asignarle valores a sus propiedades.

Un ejemplo sencillo podría ser el siguiente:

```


   public String cambiar() {


      //Creamos un bean en la request externo

      // al bean actual

      Ejemplo6b ejemplo6b = (Ejemplo6b) getBean("ejemplo6b");


      //fijamos propiedades de este bean
```

```
        ejemplo6b.setNombre("un nuevo nombre");

        //Cuando el controlador navege al jsp de salida ejemplo6b
        // no construirá un nuevo bean, ya que existirá en
        // la request
        return "ejemplo6b";
    }
```

Donde suponemos que tenemos dos páginas ejemplo6.jsp y ejemplo6b.jsp, cada una de ellas con el backbean ejemplo6 o ejemplo6b, que son objetos de clase Ejemplo6 y Ejemplo6b que se han declarado en el faces-config.xml así:

```
    <!-- managed beans para ejemplo6 -->
    <managed-bean>
        <managed-bean-name>ejemplo6</managed-bean-name>
        <managed-bean-class>
            adictosaltrabajo.ejemplo6.Ejemplo6
        </managed-bean-class>
        <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>
    <managed-bean>
        <managed-bean-name>ejemplo6b</managed-bean-name>
        <managed-bean-class>
            adictosaltrabajo.ejemplo6.Ejemplo6b
        </managed-bean-class>
        <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>
```

Cómo seguir la pista a una petición JSF

El punto más oscuro cuanto de enfrentas por primera vez a una página JSF es determinar qué código se está ejecutando y qué elementos intervienen en tu petición. Vamos a escribir unas sencillas reglas que nos ayudarán a seguir el rastro de la petición, y saber qué se está ejecutando. Hay dos tipos de peticiones importantes en JSF:

- navegación a una url con extensión JSF (normalmente *.jsf o *.faces). Este es el caso cuando escribimos en la URL del navegador una url que está controlada por JSF, por ejemplo siguiento el enlace de una página que estamos visitando.

- Ejecución de una acción dentro de una página JSF. Este es el caso que se produce cuando pinchamos en un botón de una página de una aplicación JSF.

Seguir la pista a peticiones .jsf

Este es el caso más sencillo. Partimos de la URL que queremos investigar. En este caso la url nos da la pista de la página JSP que se ha cargado, ya que cada url de JSF debe tener asociada su correspondiente JSP. Analizando las expresiones EL de la página JSP encontraremos los beans que se cargan para construir la página, y el órden en que se llaman a sus propiedades.

Seguir la pista a acciones JSF

Este es el caso más complejo. Partimos de una página que estamos visualizando y sabemos que se ha pulsado un botón. Lo primero que hay que tener en cuenta es que la URL que aparece en la barra del navegador no implica que sepamos cual es la JSP asociada a la pagína que estamos viendo. En efecto, si se ha producido una navegación, la URL que aparece es la origen de la petición, pero no la de destino.

Por tanto, para seguir la pista a una acción primero vamos a suponer que estamos en el caso en el que la JSP que estamos visualizando se corresponde con la URL de JSF que vemos en la barra del navegador. Para seguir la pista a la acción de un botón de esa página seguimos el siguiente método:

1. Buscamos en la página JSP el código del botón. Por ejemplo podría ser algo así:

```
<h:commandButton action="#{ejemplo3.enviar}" value="Enviar success o error" />
```

2. De la expresión EL de la acción del botón obtenemos el nombre del bean y el nombre del método que ejecuta la acción. Normalmente las acciones serán del tipo #{beanName.actionName}, por lo que el nombre del bean sería beanName y el del método que ejecutará la acción actionName. En el ejemplo amterior, el bean es ejemplo3 y el método que se ejecutará es enviar.

3. Ahora buscamos la clase del bean. Para ello buscamos en el faces-config.xml el managed-bean que hemos encontrado. En nuestro ejemplo la sección correspondiente indica, como vemos a continuación, que el bean ejemplo3 se crea en el ámbito request de la petición y es de clase: `adictosaltrabajo.ejemplo3.Ejemplo3`

```
<!-- managed beans para ejemplo3 -->
<managed-bean>
    <managed-bean-name>ejemplo3</managed-bean-name>
    <managed-bean-clas>
        adictosaltrabajo.ejemplo3.Ejemplo3
    </managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

4. Ahora localizamos en la clase Ejemplo3 el método enviar, que será un public String enviar()

```
public class Ejemplo3 {
    ...
```

```
    public String enviar() {

        //aquí irá la lógica de negocio de esta acción

        return "success";
    }
```

5. Tras seguir la pista a la lógica de negocio que veremos en el método (el de ejemplo es muy simple, no parace ninguna), nos fijamos en las sentencias return del método, ya que son las que generan el valor de retorno del método. En nuestro ejemplo también es muy simple, ya que sólo contiene el return “siccess”.

6. Cuando hemos localizado la cadena de retorno, en nuestro casso “success”, buscamos en el faces-config.xml la regla de navegación cuya entrada sea el JSP inicial. En nuestro caso las reglas de navegación para esta petición son:

```
    <!-- Reglas de navegacion para el ejemplo 3-->
    <navigation-rule>
        <from-view-id>/ejemplo3/ejemplo3.jsp</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>
            <to-view-id>/ejemplo3/panelSuccess.jsp</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>OK</from-outcome>
            <to-view-id>/ejemplo3/panelOk.jsp</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>error</from-outcome>
            <to-view-id>/ejemplo3/panelError.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
```

7. la etiqueta from-view-id nos dice las JSP de entrada que serán controladas por esta regla de navegación (puede contener caracteres comodín). Las etiquetas from-outcome nos dan los posibles valores de las cadenas retornadas por las acciones, y las etiquetas to-view-id nos dan las JSP de salida correspondientes. Estas dos etiquetas irán siempre en pares de etiquetas dentro de una etiqueta navigation-rule, que define una regla de navegación. En nuestro ejemplo la cadena "success" lleva a la página `/ejemplo3/panelSuccess.jsp`

8. Ahora ya sabemos la página por la que saldrá la acción. Sólo nos resta echarle un vistazo a dicha página para saber los beaqns que se van a necesitar para su construcción.

En el caso más complicado, la página que estamos viendo la hemos obtenido tras ejecutar varias acciones. Esto nos obligará a repetir el proceso anterior varias veces desde el inicio de la petición para saber a qué pagina hemos llegado. Para evitarnos estos líos conviene porner al principio de la página algún comentario en el propio código fuente de la página que nos indique el JSP que estamos visualizando.

JSF frente a Struts

JSF es uno de los posobles frameworks que se pueden utiliar para crear aplicaciones web Java. Uno de los más extendidos es Struts, un framework bien maduro, que sin embargo no forma parte del estándar J2EE.

En esencia, Struts es un controlador MVC de tipo 2, en el que la acción del formulario se liga a una clase acción, que sigle el patrón "comando" de diseño (una clase que representa un comando). El controlador de Struts sigue a su vez el patron de "controlador frontal", dirigiendo cada petición solicitada a la clase acción qie la procesa, y paándole los datos de la petición encapsulados en un objeto (patrón "objeto valor").

JSF se desarrolló teniendo en mente Struts, ya que los impulsores de este desarrollo, en los que se encontraban gente como IBM y el propio creador de Struts, querían crear un nuevo framework más potente.

Prácticamente todo lo que se puede hacer con Struts tiene un equivalente JSF (hay como siempre que adaptarse al modo de trabajo de JSF, no intentar "clonar" el comportamiento de Struts en JSF), pero además utilizando JSF obtendremos una serie de ventajas:

- La arquitectura del controlador es más avanzada y flexible. Podremos hacer algunas traseas avanzadas de manera sencilla (por ejemplo, utilizando "phase listeners").
- JSF permite definir la navegación no solo a través de los métodos de navegación de los beans, sino incluso en la propia página (navegación definida en los componentes de la página).
- JSF permite recoger los parámetros del formulario de manera más sencilla que Struts, e incorpora un lenguaje de expresiones que lo hace más simple.
- JSF soporta la creación de manejadores de eventos asociados a los componentes de la página, lo que dota a dichos componentes de gran potencia. Un ejemplo: creación de combos enlazados, en los que la elección de un elemento en el primer combo obliga a recalcular los elementos disponibles en el segundo combo, por ejemplo, en ombos de paises y estados o ciudades.
- JSF está pensado para la creación de interfaces de usuario avanzadas. Basta ver los nuevos frameworks extendidos JSF, como Apache MyFaces Trinnidad y el resto de frameworks JSF con soporte AJAX.
- Em resumen, para los desarrolladores que han usado algún lenguajje visual (visual basic, delphi, swing, etc.) el modo de trabajo de JSF pronto les parecerá cercano.

Para un proyecto nuevo, la elección es sencilla, como dice el propio creador de Struts: utiliza JSF.

JSF y AJAX

JSF es un framework que lanza muchas peticiones al servidor. Para optimizar dicho dialogo eftan clmentando a aparecer implementaciones de JSF que incorporan AJAX en sus etiquetas. Esto permite actualitar los fcomponentes en el navegador del usuario de manera selectiva, sin necesidad de recargar la página completa. La comyinación JSF ajax dpta a las paginas de gran dinamismo sin complicar el desarrollo, eiitando el uso de juvascript codificado a mano1 asegurando un mayor soporte alos navegadores web.

El futuro de JSF

El framework JSF forma parte importante del estándar java J2EE. De hecho se está preparando una nueva versión que taerá numerosas novedades, sobre todo en lo que se refiere a su integración con AJAX. También se está comenzando a utilizar en numerosas aplicaciones empresariales, ya que permite crear pantallas de usuario bastante complejas con uina cierta facilidad, aunque desde luego no es sencillo la primera vez que te enfrentas a este framework. En la nueva versión se espera una mejora sobre el control de las fases del ciclo de vida de la petición que faciliten la creación de componentes JSF complejos que se usan de manera simple.

En un artículo posterior intentaré poneror ejemplos de aplicaciones basadas en JSF y AJAX utilizando alguno de los frameworks más importantes, como puede ser MyFaces Trinidad o IceFaces.

Conclusión

Este artículo sirve de introducción al framework JSF. Será complementado con otro artículo que incluirá una aplicación de ejemplo lista para funcionar que explicará muchas de las técnicas funcamentales de creación de aplicaciones JSF. Espero que con este artículo y echando un vistazo a aplicaciones JSF ya desarrolladas os podais hacer una idea de sus posibles aplicaciones, así como de su potencia.

Un consejo, una vez que comenceis a desarrollar una apliación de gran envergadura con JSF os dareis rápidamente cuenta que con el API JSF básico no vais muy lejos. En mi caso llevo bastante tiempo usando MyFaces Tomahawk y la verdad es que es bastante potente, eliminando bastantes de las restricciones que impone el JSF Çore Tags, y que añade etiquetas bastante útkiles.

Votar

- Puedes opinar sobre este tutorial 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

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

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subcripción a novedades:

E-mail Aceptar

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Arquetipos de maven: cómo crear, distribuir y generar proyectos con JSF e ICEfaces, JBoss y EJB3	Maven permite la creación de arquetipos de manera muy flexible. En este tutorial veremos cómo crear y distribuir uno que integre las tecnologías JSF e IceFaces, JBoss y EJB3	2008-06-09	3274	Muy bueno	4	
Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: aplicación multimódulo	En este artículo se va a abordar el desarrollo de una aplicación Myfaces JSF con Maven que sea multimódulo.	2007-07-11	5943	Muy bueno	3	
Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: pruebas con Jetty y Tomcat	Este es el tercer tutorial de la "saga" de Maven, JSF y Eclipse, donde se va a realizar las pruebas de la aplicación sobre dos servidores web diferentes: el servidor Jetty, integrado en Maven, y el servidor Tomcat, que lo integraremos con Eclipse.	2007-09-10	7447	Bueno	1	
Pruebas unitarias Web para aplicaciones JSF	En este tutorial se puede encontrar una introducción y un análisis de los diferentes frameworks disponibles para realizar pruebas unitarias web de aplicaciones JSF	2006-11-13	7061	Bueno	1	
Proyecto con JSF Myfaces, Maven y Eclipse	En este tutorial vamos a aprender a construir una aplicación básica JSF (Java Server Pages) utilizando el Maven 2.0 y las bibliotecas de MyFaces. Lo mejor de todo es que para crear el ejemplo no vamos a programar ni una línea.	2007-05-28	12738	Bueno	7	
Guía de referencia de JSF	Esta guía-tutorial pretende dar a conocer todos los conceptos básicos de JSF así como servir de guía de referencia a los desarrolladores.En ningún caso esta pensada para aprender JSF desde cero.	2007-02-09	16748	Bueno	7	
Introducción a Ajax4Jsf	En este tutorial se hablará de Ajax4jsf, una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript.	2007-04-09	14388	Bueno	6	
Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6	Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6	2007-12-10	7544	Regular	5	
Como hacer un componente de JSF	En este tutorial Alejandro Pérez nos enseñará como construir nuestro propio componente en JSF mediante un ejemplo	2007-07-05	7472	Regular	6	
Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: Hibernate (segunda parte)	En este artículo se va a continuar con el desarrollo de la aplicación Myfaces JSF con Maven multimódulo que comenzamos en un tutorial anterior. Además también se tratará de la integración de Hibernate con las aplicaciones.	2007-07-31	7445	Malo	1	

Nota: