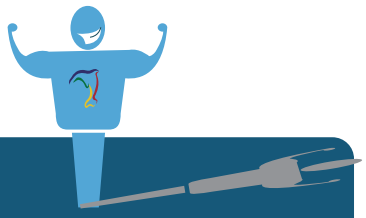
autentia

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...

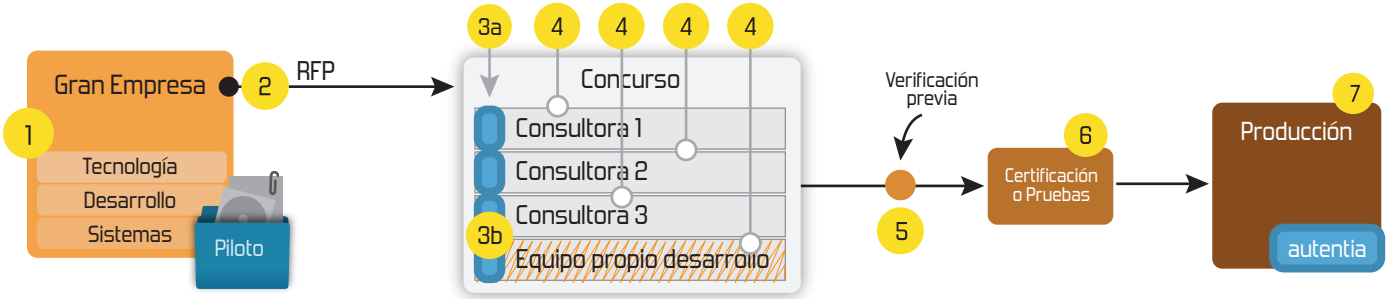
1. Desarrollo de componentes y proyectos a medida



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

Lanzado TNTConcept versión 0.8 (10/12/2007)

¿Gestionas tu empresa con hojas de cálculo? ¿No crees que puede haber un modo mejor?

Desde Autentia ponemos a vuestra disposición el software que hemos construido (100% gratuito, con código fuente disponible y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (au**T**e**NT**ia). Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño ..... Saber más en:
http://tntconcept.sourceforge.net/

Tutorial desarrollado por: Iván Zaera Avellón

Puedes encontrarme en Autentia
Somos expertos en Java/J2EE
Contacta en:
izaera@autentia.com

NUEVO CATÁLOGO DE SERVICIOS DE AUTENTIA (PDF 6,2MB)

www.adictosaltrabajo.com es Web de difusión de conocimiento de www.autentia.com

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF tomcat6_icefaces.pdf

Firma en nuestro libro de Visitas **<-----> Asociarme al grupo AdictosAlTrabajo en eConozco**

Trabaje desde casa
Ingresos extra desde casa. Tiempo completo o parcial.
www.trabajaportucuenta.com
Anuncios Google - Anunciarse en este sitio

Fecha de creación del tutorial: 2007-12-10

Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6

Introducción

En este tutorial vamos a ver como configurar lo ultimísimo en desarrollo de aplicaciones web. Primeramente vamos a ver en qué para hacer esta afirmación. Posteriormente, veremos que requisitos debe cumplir un proyecto para qué funcione. Ya veremos qu buen funcionamiento depende mucho de las versiones de las librerías e incluso de algún fichero de configuración. Como herrami compilación propondremos Maven.

Este tutorial asume conocimientos de uso de JSF, Facelets y Maven. Si no es así, por favor, consulta los tutoriales ya existentes AdictosAlTrabajo sobre estos temas.

Opinión sobre JSF 1.2+Facelets

Como he dicho en la introducción, considero la integración de estas tecnologías lo ultimísimo en desarrollo de aplicaciones web. dicho "aplicaciones web", no "portales web" ni "gestores de contenido". Que es de lo último que ha salido es un hecho objetivo, palabra "ultimísimo" lleva asociado un matiz como de excelencia o superioridad sobre otras tecnologías. ésto es claramente una subjetiva mía que trataré de argumentar a continuación.

Todos los que llevamos unos pocos años desarrollando en Java sabemos los altibajos que ha tenido la plataforma y, muy especi la crisis en la que parecíamos sumidos los últimos años. Al menos, yo empezaba a rozar la desesperación, al ver lo frustrante y productivo que iba siendo el desarrollo. En especial, el tema de la productividad empezaba a ser preocupante, por la gran cantid ficheros XML que había que configurar y lo difícil que era desplegar algo para que funcionase, o peor aun, se mantuviese funcior Cada vez había que dedicar más tiempo a la tecnología y menos al análisis, diseño y codificación. O sea, que trabajábamos nosc

la tecnología, en lugar de la tecnología para nosotros. Para colmo, "aparecían" nuevas herramientas, como Ruby, o .Net, que da sensación de ser mucho mas sencillas y fáciles de utilizar.

Después de muchas propuestas de frameworks, librerías, tecnologías, etc., (muchas de las cuales hemos usado en Autentia; ver tutoriales en este mismo web) se empezó a ver la luz con EJB 3.0. Parecía que, por fin, se inventaba algo en Java que era más s que lo anterior y que facilitaba la vida al desarrollador. y efectivamente, era así. Con Java 5 (anotaciones) y EJB 3.0 tenemos po forma sencilla de programar sin necesidad de miles de ficheros XML y complicados despliegues. Basta echar un JAR en JBOSS y funciona perfectamente.

A continuación seguimos con JSF 1.1, utilizando la implementación de Apache (myfaces). La idea era buena, pero el funcionamie y complicado. Muchas cosas parecían contraintuitivas, y la curva de aprendizaje era muy alta. Sin embargo, la idea de modelar aplicaciones web como se ha hecho toda la vida en entornos de escritorio (Windows, GTK, QT, Swing, ...) prometía. En cualquier algo faltaba, los JSPs se habían quedado cortors (eran un poco engorrosos) y no encajaban bien con JSF.

Es ahora cuando, por fin, con la llegada de JSF 1.2 y Facelets, podemos desarrollar aplicaciones fácilmente, como lo hacíamos e MFCs (Windows): con un modelo de aplicación basado en componentes, donde la pantalla se ve como un árbol de controles (bot listados, etc.) reales, y no como una pagina HTML donde embutimos cosas que al usuario le parecen controles pero que para no los desarrolladores, son parámetros de una petición HTTP y cosas que nada tienen que ver con lo que el usuario está acostumbr manejar.

Ya veréis, ya. Echarle un vistazo, probad un poquito, y no querréis volver a oír hablar de JSPs, Struts, GWT, o demás framework es así, siempre podéis dejarme un comentario en este tutorial (en esta página, abajo del todo) poniéndome a caldo. Yo, desde l pienso recomendar a todo el que me pregunte que use estas tecnologías si no quiere tener que migrar sus aplicaciones dentro d año o dos.

Es estándar y es el futuro. No cabe duda.

Opinión sobre ICEFaces

Separo ICEFaces de JSF 1.2+Facelets porque los dos últimos son estándar, mientras que ICEFaces no. ¿Por qué ICEFaces? Desd empezó a hablar de AJAX, han surgido multitud de frameworks (Prototype, DWR, GWT, ...) que, si bien aportaban facilidad de u acababan de convencer. Algunos porque sólo eran Javascript cliente, otros porque, si bien integraban la parte de servidor con la cliente, no eran realmente frameworks, sino meras librerías de comunicación. Además, no estaba claro como juntarlos con la ar JEE.

Con la llegada de JSF, se empezaron a vislumbrar posibilidades de integración. Si JSF permitía al desarrollador aislarse de la arq web y ver sus aplicaciones como algo parecido a una aplicación de escritorio, debería entonces ser sencillo utilizar AJAX para ha controles más funcionales. Y así fue, empezaron a aparecer AJAX4JSF, ICEFaces, Tobago, ...

Sin embargo, de estas propuestas, ICEFaces es la que más me gusta, porque aisla completamente al desarrollador de AJAX. No falta etiquetas especiales: se ponen los controles en la pantalla e ICEFaces se encarga de enviar entre cliente y servidor sólo la información necesaria. Es decir, ya no se envían los formularios a la antigua usanza, en un POST de HTTP, sino que solo se envía cambios que ha hecho el usuario del cliente al servidor, y los cambios en la pantalla del servidor al cliente. No mas parpadeos en pantalla y pantalla, no mas sobrecarga de red, no más aplicaciones pesadas. Además, con la inclusión de la librería Scriptaculou ICEFaces, se dispone de arrastrar+soltar y de efectos (fundidos, parpadeos, apariciones, ...) para los controles.

Una maravilla.

Configurar el proyecto

Por lo que llevamos investigado hasta el momento en Autentia, montar un proyecto con estas tres tecnologías sobre JBoss es al relativamente sencillo. Sin embargo, si queremos ejecutar en Tomcat 6, la cosa cambia. ésto es debido a que este despliegue es dependiente de las versiones de las librerías. Por ello, vamos a ver que "pom.xml" (fichero de configuración de Maven) necesitar proyecto para funcionar:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_

        <modelVersion>4.0.0</modelVersion>
        <groupId>com.autentia</groupId>
        <artifactId>tutorial</artifactId>
        <packaging>war</packaging>
        <version>1.0-SNAPSHOT</version>

        <dependencies>

                <!-- Core -->
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                        <version>1.2.14</version>
                </dependency>
```

```
<!-- Core J2EE -->
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.3</version>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.1.2</version>
</dependency>
<dependency>
        <groupId>taglibs</groupId>
        <artifactId>standard</artifactId>
        <version>1.1.2</version>
</dependency>
<dependency>
        <groupId>javax.el</groupId>
        <artifactId>el-api</artifactId>
        <version>1.0</version>
        <scope>provided</scope>
 </dependency>

<!-- JSF -->
<dependency>
        <groupId>javax.faces</groupId>
        <artifactId>jsf-api</artifactId>
        <version>1.2_04-p01</version>
</dependency>
<dependency>
        <groupId>javax.faces</groupId>
        <artifactId>jsf-impl</artifactId>
        <version>1.2_04-p01</version>
</dependency>
<dependency>
        <groupId>org.icefaces</groupId>
        <artifactId>icefaces-facelets</artifactId>
        <version>1.6.1</version>
</dependency>

<!-- ICEFaces -->
<dependency>
        <groupId>org.icefaces</groupId>
        <artifactId>icefaces</artifactId>
        <version>1.6.1</version>
</dependency>
<dependency>
        <groupId>org.icefaces</groupId>
        <artifactId>icefaces-comps</artifactId>
        <version>1.6.1</version>
</dependency>
<dependency>
        <groupId>backport-util-concurrent</groupId>
        <artifactId>backport-util-concurrent</artifactId>
        <version>3.1</version>
</dependency>
<dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1</version>
</dependency>
<dependency>
        <groupId>commons-beanutils</groupId>
        <artifactId>commons-beanutils</artifactId>
        <version>1.7.0</version>
</dependency>
<dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2</version>
</dependency>
<dependency>
        <groupId>commons-digester</groupId>
        <artifactId>commons-digester</artifactId>
        <version>1.8</version>
</dependency>
<dependency>
        <groupId>commons-fileupload</groupId>
```

```
                        <artifactId>commons-fileupload</artifactId>
                        <version>1.2</version>
                    </dependency>
                    <dependency>
                        <groupId>commons-discovery</groupId>
                        <artifactId>commons-discovery</artifactId>
                        <version>0.4</version>
                    </dependency>
                    <dependency>
                        <groupId>commons-el</groupId>
                        <artifactId>commons-el</artifactId>
                        <version>1.0</version>
                    </dependency>
                    <dependency>
                        <groupId>xerces</groupId>
                        <artifactId>xercesImpl</artifactId>
                        <version>2.8.1</version>
                    </dependency>
                    <dependency>
                        <groupId>net.sf.jcharts</groupId>
                        <artifactId>krysalis-jCharts</artifactId>
                        <version>1.0.0-alpha-1</version>
                    </dependency>

                </dependencies>

                <repositories>
                    <repository>
                        <id>java.net</id>
                        <url>http://download.java.net/maven/1</url>
                        <layout>legacy</layout>
                    </repository>
                </repositories>

            </project>
```

Cosas importantes a tener en cuenta sobre este pom:

- Debéis añadir el repositorio Maven de Sun (http://download.java.net/maven/1) para encontrar sus librerías.
- Las dependencias de ICEFaces, deben ser descargadas manualmente de http://www.icefaces.org e instaladas en el reposi local.
- La dependencia "el-api" debe ser "provided". Si echáis un "el-api.jar" en el directorio "WEB-INF/lib" de vuestra aplicación, no arrancará.
- La dependencia "icefaces-facelets" proporciona la implementación de Facelets. Por lo tanto, no añadáis ninguna otra depe de ninguna implementación de facelets o tendréis conflictos entre ambas y no funcionará.
- Las versiones del API y de la implementación de JSF deben ser las que se especifican en el "pom" (1.2_04-p01) y no otras otras no funciona bien, dando una excepción "Expired View" (la vista ha expirado). Por supuesto, ésto es sólo válido para 1.6.1 de ICEFaces.

Si buscáis en los foros, existe un repositorio Maven en el sitio web de JBoss con la dependencia ICEFaces configurada para evita especificar las dependencias transitivas. ¡No la uséis si vais a desplegar en Tomcat porque no funciona!

Para finalizar, citaremos otros puntos a tener en cuenta para este despliegue, que van más allá del "pom". Estos atañen a los fic configuración de la aplicación web:

web.xml

Hay que tener definidos los siguientes servlets:

```
<servlet>
            <description>Servlet JSF estandar (para usar sin ICEFaces)</description>
            <servlet-name>Faces Servlet</servlet-name>
            <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
            <description>Servlet JSF de ICEFaces</description>
            <servlet-name>Persistent Faces Servlet</servlet-name>
            <servlet-class>com.icesoft.faces.webapp.xmlhttp.PersistentFacesServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
            <description>ICEFaces request blocking servlet</description>
            <servlet-name>Blocking Servlet</servlet-name>
```

```
            <servlet-class>com.icesoft.faces.webapp.xmlhttp.BlockingServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
        </servlet>
```

Y los mapeos de servlets:

```
    <servlet-mapping>
            <servlet-name>Persistent Faces Servlet</servlet-name>
            <url-pattern>*.jsf</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
            <servlet-name>Persistent Faces Servlet</servlet-name>
            <url-pattern>/xmlhttp/*</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
            <servlet-name>Blocking Servlet</servlet-name>
            <url-pattern>/block/*</url-pattern>
        </servlet-mapping>
```

Fijaos en que la extensión .JSF la mapeamos al servlet de ICEFaces (Persistent Faces Servlet), no al de la implementación de re de JSF (Faces Servlet).

faces-config.xml

Lo primero de todo, es configurar el "faces-config.xml" con el esquema XML de la versión 1.1 de JSF. ésto hace que ICEFaces se en modo compatibilidad 1.1 en JSF 1.2. Si no fuera así no funcionaria, porque ICEFaces no funcionará en modo JSF 1.2 puro has versión 2.0. Para ello hay que definir el siguiente DOCTYPE de XML:

```
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
                              "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
```

Y añadir el namespace por defecto siguiente al elemento "faces-config":

```
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
```

Desgraciadamente, al ejecutarse en modo compatibilidad con JSF 1.1, algunas de las novedades de JSF 1.2 (como por ejemplo, tener que usar la etiqueta "f:loadBundle" en cada página para cargar los recursos multi-idioma) no estarán disponibles. Por suer algo pasajero.

A continuación debemos añadir el soporte de Facelets:

```
<application>

    ...

    <view-handler>com.icesoft.faces.facelets.D2DFaceletViewHandler</view-handler>

    ...

</application>
```

Y ya tenemos nuestro proyecto listo para funcionar en Tomcat 6.0.

Conclusiones

Como veis, no es difícil integrar todas estas tecnologías en un solo proyecto. Sin embargo, hay que ser muy cuidadoso con las v y la configuración de los proyectos porque puede impactar mucho en su funcionamiento.

En Autentia contamos con una amplia experiencia en desarrollo de aplicaciones, por lo que siempre recomendamos a nuestros c enfocarse en la resolución de la problemática del negocio y delegar las partes más complicadas de la tecnología a empresas especializadas. Ya sabéis que siempre nos tenéis disponibles para ello en http://www.autentia.com.

Disfrutad del futuro del desarrollo de aplicaciones web, que ya esta aquí.

SOME RIGHTS RESERVED This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License.
Puedes opinar sobre este tutorial aquí

Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿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?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....
Autentia = Soporte a Desarrollo & Formación

Autentia S.L. Somos expertos en:
J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto **Descripción**

Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

Patrocinados por enredados.com .... Hosting en Castellano con soporte Java/J2EE