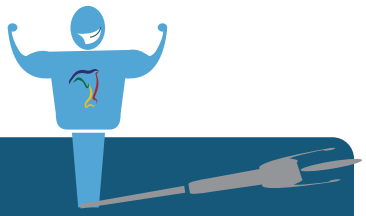
autentia

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...

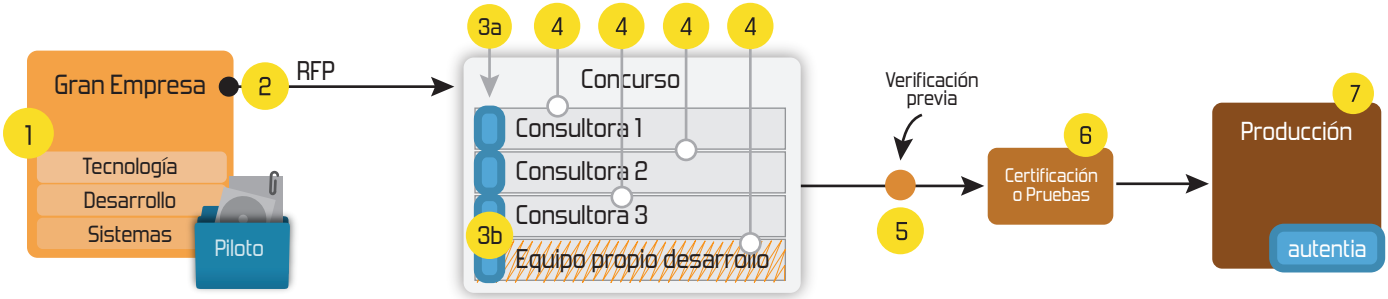
1. Desarrollo de componentes y proyectos a medida



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

Hosting Patrocinado por enREDados.com

Home | Quienes Somos | Empleo | Tutoriales | Contacte

Ver cursos que ofrece Autentia - - Descargar comics en PDF y alta resolución

Tutorial desarrollado por:

Francisco Javier Martínez Páez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Nuevo catálogo de servicios de Autentia

Descargar catálogo (6,2 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 IceJboss2.pdf

Trabaje desde casa
Ingresos extra desde casa. Tiempo completo o parcial. www.trabajaportucuenta.com

Anuncios Google

Fecha de creación del tutorial: 0000-00-00

Icefaces, JBoss, Maven2 y EJB3: Parte 2.

Como ya os prometí en el anterior tutorial de esta 'saga', en este pretendemos configurar la parte del módulo Web y del módulo Modelo para empezar a trabajar.

Por lo tanto, nuestro objetivo será configurar correctamente la aplicación Web para que se pueda desplegar correctamente en Jboss e Icefaces y configurar el proyecto eclipse de manera correcta al igual que el proyecto EJB.

Suponemos que ya tenéis instalado JBoss 4.2.2 y 'enlazado' con eclipse y habéis finalizado con éxito el tutorial anterior.

1. **Estructura del proyecto WEB.**

   Al ser un proyecto Web dinámico, y siguiendo la estructura de Maven2, tendremos: (si no lo tenéis así, modificadlo para que así sea)

   - Los fuentes en la carpeta src/main/java
   - Los recursos en src/main/resources (No olvidéis añadirla al proyecto como carpeta de fuentes)
   - Las clases compiladas en: target/classes
   - Los ficheros de la aplicación Web: src/main/webapp

2. **Creando descriptor web.xml.**

El descriptor web.xml (en src/main/webapp/WEB-INF) tendrá la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
<display-name>Web</display-name>
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>Faces Servlet</servlet-name>
<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<context-param>
<param-name>javax.faces.STATE_SAVING_METHOD</param-name>
<param-value>server</param-value>
</context-param>
<context-param>
<param-name>com.icesoft.faces.uploadDirectory</param-name>
<param-value>upload</param-value>
</context-param>
<context-param>
<param-name>com.icesoft.faces.uploadMaxFileSize</param-name>
<param-value>4048576</param-value>
</context-param>
<servlet>
```

```
<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>
<servlet-name>Blocking Servlet</servlet-name>
<servlet-class>com.icesoft.faces.webapp.xmlhttp.BlockingServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>*.jspx</url-pattern>
</servlet-mapping>
<!-- Persistent Faces Servlet Mapping -->
<servlet-mapping>
<servlet-name>Persistent Faces Servlet</servlet-name>
<url-pattern>*.iface</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>
<servlet-mapping>
<servlet-name>Blocking Servlet</servlet-name>
<url-pattern>/block/*</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>uploadServlet</servlet-name>
```

```
<servlet-class>com.icesoft.faces.component.inputfile.FileUploadServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>uploadServlet</servlet-name>
<url-pattern>/uploadHtml</url-pattern>
</servlet-mapping>
<listener>
<listener-class>com.icesoft.faces.util.event.servlet.ContextEventRepeater</listener-class>
</listener>
</web-app>
```

3. **Fichero de configuración de JSF: faces-config.xml**

El fichero de configuración faces-config.xml (en src/main/webapp/WEB-INF) tendrá la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC
"-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
"http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config>
<managed-bean>
<description>
Test Bean
</description>
<managed-bean-name>testCtrl</managed-bean-name>
<managed-bean-class>
com.autentia.tutoriales.icefaces.beans.TestCtrl
</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
</faces-config>
```

Por ahora, sólo hemos configurado un bean de Sesión que nos servirá de ejemplo.

4. **Fichero index.jsp y test.jspx**

   Crearemos dos ficheros jsp. El primero, que será el fichero de bienvenida, que redirigirá al fichero test.jspx. Nuestro único objetivo será demostrar que hemos configurado todo correctamente.

   El fichero index.jsp (en src/main/webapp) tendrá la siguiente forma:

```
<html>
```

```
<head>
</head>
<body>
<jsp:forward page="test.iface" />
</body>
</html>
```

El fichero test.jspx (JSP y XML) (en src/main/webapp) tendrá la siguiente forma:

```
<f:view xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ice="http://www.icesoft.com/icefaces/component"
xmlns:jsp="http://java.sun.com/JSP/Page">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></meta>
<title>Test</title>
</head>
<body>
<ice:outputLabel for="" value="#{testCtrl.label}"></ice:outputLabel>
</body>
</html>
</f:view>
```

4. **Vamos a crear el Bean.**

Crearemos la clase: com.autentia.tutoriales.icefaces.beans.TestCtrl:

```
package com.autentia.tutoriales.icefaces.beans;

public class TestCtrl {
public String getLabel() {
return "Esta es una etiqueta";
}
}
```

6. **Configuración de las trazas.**

Cread en la carpeta src/main/resources los ficheros:

- commons-logging.properties

```
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
```

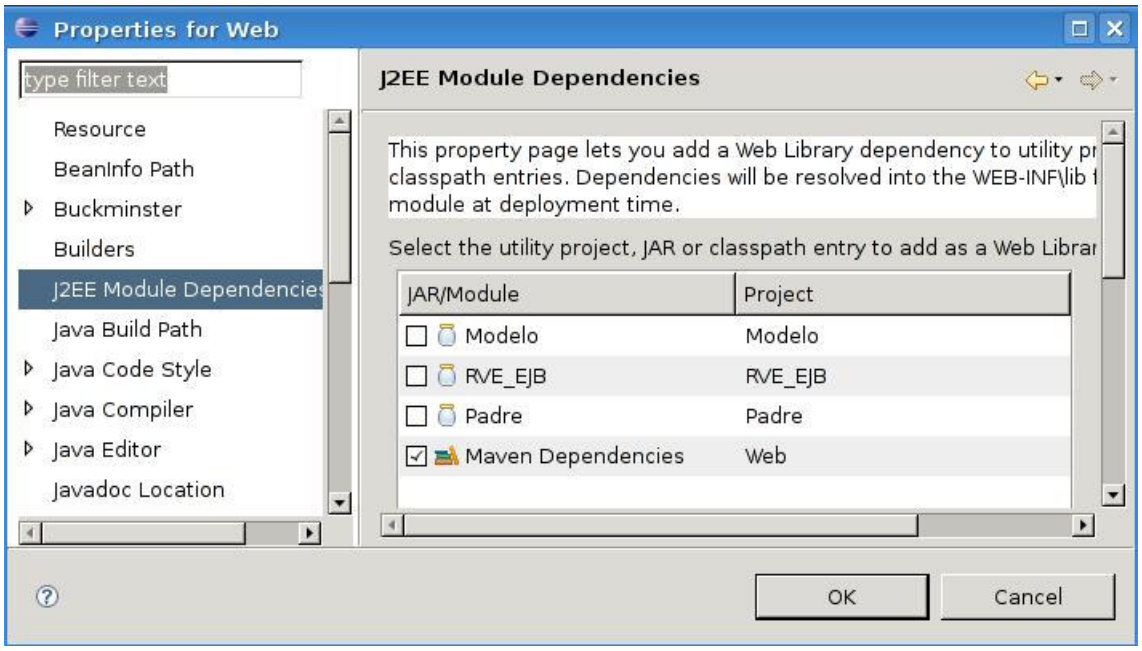
- log4j.properties

```
# Main loggers

log4j.rootLogger=INFO, stdout

log4j.logger.org.apache=ERROR


# Console appender

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1} - %m%n
```

7. **Preparando para desplegar el proyecto WEB**

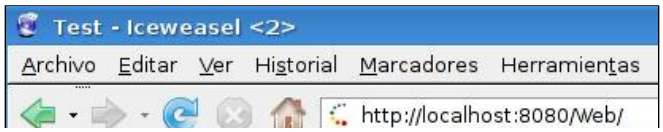
Lo añadimos al servidor Jboss en eclipse (no lo arranquéis aún)

Pulsamos sobre propiedades del proyecto y en 'J2EE Module Dependencies' marcamos únicamente 'Maven Dependencies'



Arrancamos el servidor y entramos en http://localhost:8080/Web

Debemos ver una página muy simple:



8. **Estructura del proyecto Modelo.**

   Al ser un proyecto EJB 3.0, y siguiendo la estructura de Maven2, tendremos: (si no lo tenéis así, modificadlo para que así sea)

   - Los fuentes en la carpeta src/main/java
   - Los recursos en src/main/resources (No olvidéis añadirla al proyecto como carpeta de fuentes)
   - Las clases compiladas en: target/classes

9. **Configuración de las trazas.**

   Haced lo mismo que hicimos para el módulo Web

10. **Crearemos un EJB de sesión sin estado de prueba.**

    Creamos el interfaz Local: com.autentia.tutoriales.modelo.services.TestServices

```
package com.autentia.tutoriales.modelo.services;

import javax.ejb.Local;

@Local
public interface TestService {
public String test();
}
```

Y ahora el Bean: com.autentia.tutoriales.modelo.services.TestServicesImpl

```
package com.autentia.tutoriales.modelo.services;

import javax.ejb.Stateless;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

@Stateless
public class TestServiceImpl implements TestService {
private static Log log = LogFactory.getLog(TestServiceImpl.class);
public String test() {
log.info("Me llaman");
return "Y este es el valor";
}

}
```

Vamos a crearnos una clase que nos permita localizar los EJBs de manera transparente:

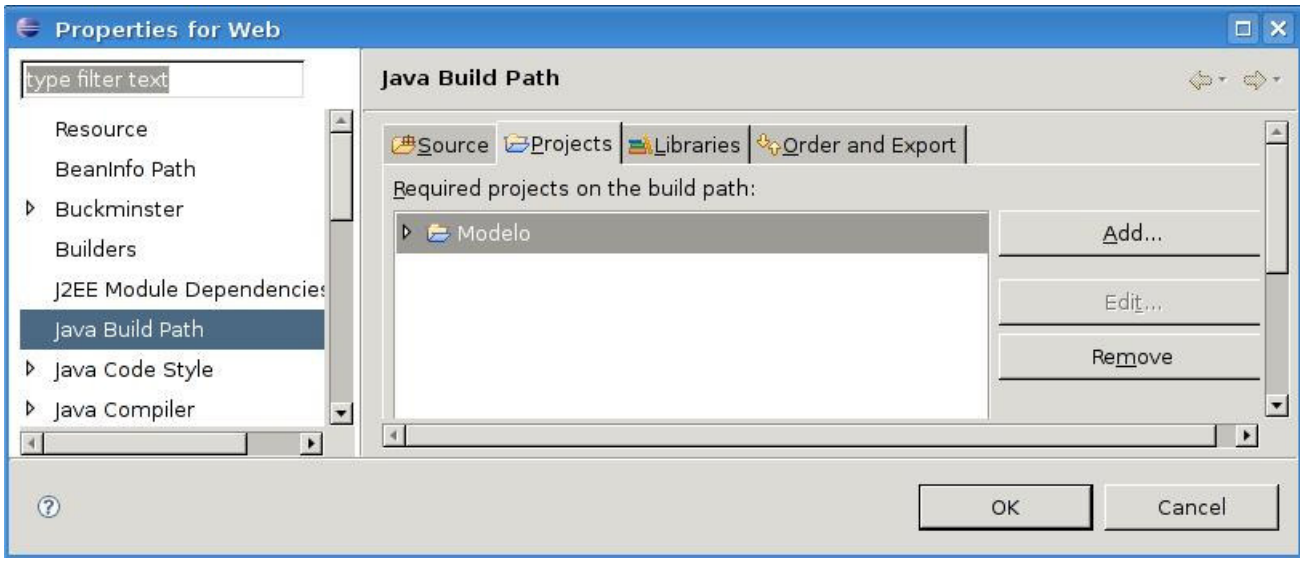
com.autentia.tutoriales.modelo.util.EjbLocator

```
package com.autentia.tutoriales.modelo.util;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

public class EjbLocator {

private static Log log = LogFactory.getLog(EjbLocator.class);

private static final InitialContext ctx;

static {

try {

ctx = new InitialContext();

} catch (NamingException e) {

log.fatal("It is not possible to create a new InitialContext.", e);

throw new RuntimeException(e);

}

}

/**

* Constructor privado.

*/

private EjbLocator() {

}

/**

* Retorna una referencia a un EJB Local

*/

@SuppressWarnings("unchecked")

public static <T> T lookupLocalBean(Class<? extends T> beanClass) throws NamingException {

final String jndiName = beanClass.getSimpleName() + "/local";

final T bean = (T)jndiLookup(jndiName);

return bean;

}

private static Object jndiLookup(String jndiName) throws NamingException {

final Object obj;

try {

obj = ctx.lookup(jndiName);

if (log.isDebugEnabled()) log.debug("Found JNDI name: " + jndiName);
```

```
} catch (NamingException e) {
log.fatal("Can not found JNDI name: " + jndiName);
throw e;
}
return obj;
}
}
```

11. **Invocamos al Modelo desde el Web:**

Configuramos el proyecto Web en eclipse añadiendo la dependencia del proyecto Modelo:



Editamos la clase TestCtrl dejándola así:

```
package com.autentia.tutoriales.icefaces.beans;

import javax.naming.NamingException;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import com.autentia.tutoriales.modelo.services.TestService;
import com.autentia.tutoriales.modelo.services.TestServiceImpl;
import com.autentia.tutoriales.modelo.util.EjbLocator;


public class TestCtrl {
private static Log log = LogFactory.getLog(TestCtrl.class);
TestService servicio;
public TestCtrl() {
log.debug("Constructor:TestCtrl");
```

```
try {
servicio = EjbLocator.lookupLocalBean(TestServiceImpl.class);
} catch (NamingException e) {
log.error ("Error al iniciar el controlador: ", e);
}
}
public String getLabel() {
log.info("Me llaman");
return "Esta es una etiqueta";
}
public String getValor() {
return servicio.test();
}
}
```

Mostramos el valor en la página test.jspx:

```
<f:view xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ice="http://www.icesoft.com/icefaces/component"
xmlns:jsp="http://java.sun.com/JSP/Page">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></meta>
<title>Test</title>
</head>
<body>
<ice:outputLabel for="valor" value="#{testCtrl.label}"></ice:outputLabel>
<ice:outputText id="valor" value="#{testCtrl.valor}"></ice:outputText>
</body>
</html>
</f:view>
```

Invocamos a la aplicación (no olvidéis añadir el proyecto Modelo al servidor y arrancarlo).: http://localhost:8080/Web/

Veremos una página muy simple:

En apariencia sólo tenemos una página muy sencilla, pero la realidad es que ya tenemos un arquetipo de aplicación bastante interesante técnicamente:

- JSF e Icefaces preparaditos para utilizar (me encanta eso del partial submit de icefaces).
- EJB 3.0 deseando ayudarnos con su magia.
- Las dependencias de las librerías manejadas por Maven2 y 'enganchadito' todo con Eclipse gracias a m2Eclipse.
- Desplegado y funcionando en Jboss.

Aún nos queda trabajo por hacer... os adelanto que lo próximo será inventarnos un modelo de datos, crearnos un esquema en MySQL que lo represente, crear las clases de entidad (Entity) y hacernos un DAO implementado en un EJB de sesión sin estado.

- 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.

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License

.

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

Formación en nuevas tecnologías

Servicio de notificaciones:

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

Formulario de subcripción a novedades:

E-mail [ ] Aceptar

Otros Tutoriales Recomendados.

Nombre corto	Descripción
EJB 3.0: Resurrection	Este turorial nos va a presentar las nuevas funcionalidades que nos aportan los EJB 3.0.
Crear el sitio web de documentación del proyecto con Maven Site	En este tutorial vamos a ver cómo crear los sitios web de documentación del proyecto con Apache Maven.

Planificar tareas en JBoss	En este tutorial os enseñaremos a planificar tareas periódicas con JBoss
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.
Despliegue gráfico de EJBs	Os mostramos como crear y desplegar de un modo gráfico un EJB de sesión el el servidor de aplicaciones de referencia de Sun
Maven, nunca antes resultó tan fácil compilar, empaquetar, ...	En este tutorial aprenderemos el uso de esta herramienta que nos permite compilar, empaquetar, generar documentación, pasar los test, preparar las builds de nuestros proyectos
Hibernate y las anotaciones de EJB 3.0	En este tutorial Alejandro Pérez nos muestra las ventajas que nos aporta Hibernate y las anotaciones de EJB 3.0
Interceptando un EJB en JBoss	En este tutorial os vamos a enseñar la arquitectura de EJBs en JBoss y a como modificarla, insertando un interceptor propio dentro de la cadena de interceptores del Proxy Cliente.
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.
Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6	Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6

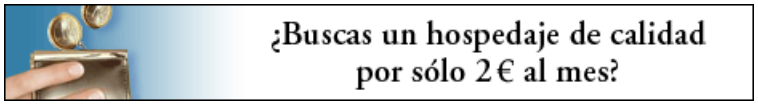
Ver todos los tutoriales

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