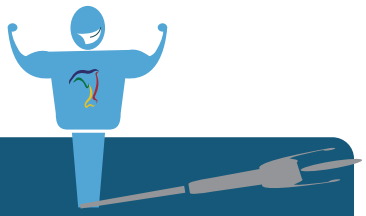
autentia

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...

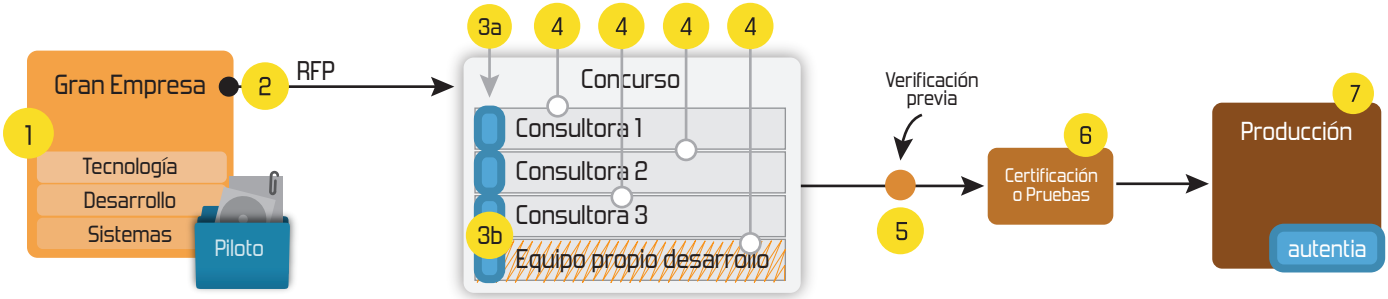
1. Desarrollo de componentes y proyectos a medida



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

Los años enseñan las cosas que los días jamás llegan a conocer
Adictos AlTrabajo
Powered by autentia
Hosting patrocinado por enredados

Ultimas Noticias

» Autentia patrocina el primer Agile Open Spain
» Nuevas funcionalidades en la web de www.adictosaltrabajo.com
» Disponible la primera versión de los plugins para integrar Maven y Bugzilla.
» Historia de la Informática. Capítulo 78. 2001-2002
» Comentando el llibro: "No he venido aquí a hacer amigos, Desventuras de un consultor IT"
» Lanzamiento del nuevo Web de Autentia
» Si se pregunta ¿Qué ofrece este Web?
» Grupo XING
» Autentia cumple 6 años

+Noticias Destacadas

» Autentia patrocina el primer Agile Open Spain
» Lanzamiento del nuevo Web de Autentia
» Autentia cumple 6 años
» Grupo XING

+Comentarios Cómic

+Enlaces

Tutorial desarrollado por

Rubén Aguilera Díaz-Heredero

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

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: MavenizarLiferaySDK.pdf

Fecha de creación del tutorial: 2009-09-27

Mavenizar Liferay SDK.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno
- 3. ¿Cómo hacer la adaptación?
- 4. Conclusiones.

1. Introducción

El desarrollo con el plugin SDK de Liferay está muy bien pero le falta algo fundamental: no tiene soporte de Maven. El soporte de Maven nos permite hacer de forma sencilla el sitio web del proyecto con informes muy útiles como: javadoc, cobertura, PMD y muchos otros informes. Además nos permite integrar de una forma muy cómoda nuestro proyecto con una aplicación de integración continua como Hudson y hacer un seguimiento de la calidad del código con Sonar.

En esta misma web tienes varios tutoriales que hablan de esta estupenda tecnología. En vez de renunciar a estas ventajas lo que podemos hacer es adaptar la creación de un proyecto con el SDK a un proyecto de Maven utilizando Eclipse.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Asus G50Vseries (Core Duo P8600 2.4GHz, 4GB RAM, 320 GB HD).
- Sistema operativo: Ubuntu 9.04 Desktop
- Mozilla Firefox 2.0.0.20
- Maven 2.2.20
- Liferay 5.2.3 (Portal + Plugins SDK)

3. ¿Cómo hacer la adaptación?

El primer paso es crear un plugin de tipo portlet de la forma que se ha visto en este tutorial: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=LiferayPluginsSDK y y que vamos a usar de referencia para crear el proyecto de Maven, a partir de ahora nos referiremos como REF_PORTLET. Para ello abrimos un terminal, nos situamos en la ruta SDK_LIFERAY/portlets y escribimos lo siguiente:

```
view plain print ?
01. ./create.sh ref "RefPortlet"
```

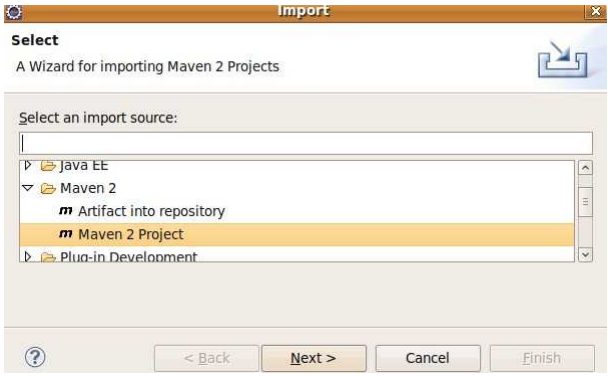
Luego tenemos que crear un proyecto Maven de tipo web, al que nos referiremos como MAVEN_PORTLET. Para ello abrimos un terminal, nos situamos en el directorio donde vayamos a crear el proyecto y ponemos lo siguiente:

```
view plain print ?
01. mvn archetype:create -DgroupId=com.autentia.training
02. -DartifactId=maven-portlet -DarchetypeArtifactId=maven-archetype-webapp
```

Para poder convertir este proyecto en un proyecto de portlet Liferay tenemos que copiar algunos archivos del proyecto de referencia a este proyecto Maven.

En primer lugar copiamos todo el contenido del directorio REF_PORTLET/docroot dentro de la carpeta MAVEN_PORTLET/src/main/webapp, sobreescribiendo todo el contenido y borramos el directorio MAVEN_PORTLET/src/main/webapp/WEB-INF/src.

El siguiente paso es importar el proyecto en Eclipse. Para ello dentro de Eclipse vamos a File→ Import → Maven 2 Project.



En la siguiente pantalla pulsamos sobre el botón "Browser", seleccionando el directorio MAVEN_PORTLET, marcamos el fichero pom.xml que detectará el asistente y pulsamos en "Finish".

Catálogo de servicios Autentia (PDF 6,2MB)

En formato comic...

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

Google

Web
www.adictosaltrabajo.com
Buscar

Últimos tutoriales

2009-09-28
iReport : Solución al problema de los parámetros dinámicos.

2009-09-27
Mavenizar Liferay SDK

2009-09-27
Crear un plugin con Liferay SDK

2009-09-23
EJB 3.0 y pruebas unitarias con Maven, JUnit 4 y Apache Open EJB.

2009-09-21
Primeros pasos con JBoss Seam

2009-09-21
Integración con sistemas de Bug Tracking desde NetBeans 6.7

2009-09-14
EJB 3.0 y pruebas de persistencia con Maven, JUnit 4 y Embedded JBoss sobre Java 6.

2009-09-12
Instalación de Liferay en Tomcat existente

2009-09-11
Release Bugzilla Maven Plugin

2009-09-11
Enlazar Bugzilla con MavenChangesPlugin

2009-09-08
Sobre las reglas de codificación o... ¿de dónde salen esos caracteres "raros"?.

2009-08-28
Cómo hacer deploy del site de Maven en SourceForge

2009-08-26
Ordenación por cantidades en informe cruzado

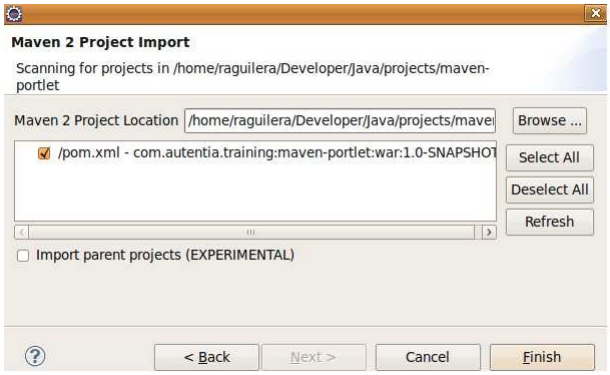
2009-08-20
Selenium IDE-Incorporando while en los test

2009-08-14
Blender y JMonkeyEngine. Exportación de archivos Blender y uso de los mismos en JMonkeyEngine

2009-08-14
5º tutorial TNT Concept Versión 0.16.1

Anuncios Google Descargar Plugin VST Coconut Fiber Import Import Bicycle Parts Portal Programas com Flash Drives Import

Copyright 2003-2009 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto



Ahora tenemos que relacionar el proyecto con el servidor que contiene Liferay. Para ello pulsamos con el botón derecho sobre la raíz del proyecto y seleccionamos la opción "Properties" → "Targeted Runtimes", seleccionamos el servidor de Liferay y pulsamos sobre "OK". De este modo ya tienen que solucionar los problemas de compilación del proyecto y podremos hacer uso de las librerías que proporciona Liferay en tiempo de compilación ya que en ejecución estas librerías ya existen.

De esta forma ya estamos en disposición de crear nuestro primer portlet. Para ello creamos una clase dentro de la carpeta src/main/java que extienda de la clase GenericPortlet como en el siguiente código:

view plain print ?

```
01. package com.autentia.training;
02.
03. import java.io.IOException;
04.
05. import javax.portlet.ActionRequest;
06. import javax.portlet.ActionResponse;
07. import javax.portlet.GenericPortlet;
08. import javax.portlet.PortletException;
09. import javax.portlet.PortletRequestDispatcher;
10. import javax.portlet.RenderRequest;
11. import javax.portlet.RenderResponse;
12.
13. import com.liferay.portal.kernel.log.Log;
14. import com.liferay.portal.kernel.log.LogFactoryUtil;
15.
16. public class MavenPortlet extends GenericPortlet {
17.
18.  public void init() throws PortletException {
19.   editJSP = getInitParameter("edit-jsp");
20.   helpJSP = getInitParameter("help-jsp");
21.   viewJSP = getInitParameter("view-jsp");
22.  }
23.
24.  public void doDispatch(RenderRequest renderRequest, RenderResponse renderResponse)throws IOException, PortletException {
25.
26.   String jspPage = renderRequest.getParameter"jspPage");
27.
28.   if(jspPage != null) {
29.     include(jspPage, renderRequest,renderResponse);
30.     }else{
31.         super.doDispatch(renderRequest,renderResponse);
32.     }
33.  }
34.
35.  public void doEdit(RenderRequest renderRequest,RenderResponse renderResponse)throws IOException, PortletException {
36.
37.   if(renderRequest.getPreferences() == null) {
38.      super.doEdit(renderRequest,renderResponse);
39.     }else{
40.      include(editJSP, renderRequest,renderResponse);
41.     }
42.  }
43.
44.  public void doHelp(RenderRequest renderRequest,RenderResponse renderResponse)throws IOException, PortletException {
45.     include(helpJSP, renderRequest, renderResponse);
46.  }
47.
48.  public void doView(RenderRequest renderRequest,RenderResponse renderResponse)throws IOException, PortletException {
49.     include(viewJSP, renderRequest, renderResponse);
50.  }
51.
52.  public void processAction(ActionRequest actionRequest,ActionResponse actionResponse)throws IOException, PortletException {
53.  }
54.
55.  protected void include(String path, RenderRequest renderRequest,RenderResponse renderResponse)throws IOException, PortletException {
56.     PortletRequestDispatcher portletRequestDispatcher = getPortletContext().getRequestDispatcher(path);
57.
58.   if (portletRequestDispatcher == null) {
59.     _log.error(path + " is not a valid include');
60.     }else{
61.         portletRequestDispatcher.include(renderRequest,renderResponse);
62.     }
63.  }
64.
65.  protected String editJSP;
66.  protected String helpJSP;
67.  protected String viewJSP;
68.
69.  private static Log _log = LogFactoryUtil.getLog(MavenPortletclass);
70. }
```

Ahora tenemos que editar el fichero de configuración portlet.xml que se encuentra en la carpeta WEB-INF del proyecto. En este fichero vamos a modificar el nombre del portlet, el nombre de visualización y sobre todo la clase. Opcionalmente también podemos modificar la información y los roles. El fichero debería presentar este aspecto:

Gestión de informes, vacaciones y utilidades

2009-08-14
Joomla 1.5. Instalación y configuración

2009-08-13
Introducción a los diagramas EPC (Event-Driven Process Chain)

2009-08-10
Blender. Animaciones avanzadas y renderización

2009-08-10
Gestión de Calidad, tablón y seguimiento en TNT Concept Versión 0.16.1

2009-08-10
Cómo hacer una página web

2009-08-06
Tips And Tricks JUnit Spring

2009-08-03
Instalación de VirtualBox PUEL

2009-08-03
Gestión de contactos y pedidos en TNT Concept versión 0.16.1

2009-08-03
Comentando el libro: La estrategia del océano azul

2009-07-30
Funciones esenciales para crear un juego.

2009-07-30
2º tutorial TNT Concept versión 1.16.1

2009-07-29
Hibernate Search, Bridges, Analizadores y más

2009-07-24
Migración de EJB3 a JPA y Spring.

2009-07-20
Directorio de ejemplos de jMonkey Engine

2009-07-19
JSR-179 Location API para J2ME: Posicionamiento geográfico en nuestras aplicaciones.

2009-07-16
Gestión de Usuarios en TNT Concept versión 0.16.1

2009-07-16
Continuación del Tutorial: JMonkeyEngine, Creación de nuestro primer juego.

2009-07-16
Como implementar el Scene Monitor para analizar las escenas en jMonkeyEngine

2009-02-26
Transformaciones de escena en JMonkeyEngine

2009-07-15
Detalles del juego de la moto en jMonekyEngine.

2009-07-14
JMonkeyEngine, Creación de nuestro primer juego.

2009-07-13
Ajax tests con Selenium: prototype.js e ICEfaces.

2009-07-08
AOP con AspectJ y Maven

2009-07-07
Instalación y configuración de Eclipse Galileo

2009-07-07
Iniciarse en el manejo de JME, Creación de un Cloth.

2009-07-06
Primeros pasos con Blender: Pintando nuestra mascota en 3D

2009-07-06
DBUnit-Exportar e Importar BBDD

2009-07-05
JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones

```
view plain print ?
<?xml version="1.0"?>
<portlet-app
xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd>
    <portlet>
        <portlet-name>maven</portlet-name>
        <display-name>Maven Portlet</display-name>
        <portlet-class>com.autentia.training.MavenPortlet</portlet-class>
        <init-param>
            <name>view-jsp</name>
            <value>/view.jsp</value>
        </init-param>
        <expiration-cache>0</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
        </supports>
        <portlet-info>
            <title>Maven Portlet</title>
            <short-title>Maven Portlet</short-title>
            <keywords>Maven Portlet</keywords>
        </portlet-info>
        <security-role-ref>
            <role-name>administrator</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>guest</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>power-user</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>user</role-name>
        </security-role-ref>
    </portlet>
</portlet-app>
```

Una vez que hemos establecido el portlet-name, pasamos a editar el fichero liferay-display.xml donde tenemos que establecer el nombre de la categoría y el id del portlet, que se corresponde con el portlet-name. El fichero debe presentar el siguiente aspecto:

```
view plain print ?
<?xml version="1.0"?>
<!DOCTYPE display PUBLIC "-//Liferay//DTD Display 5.2.0//EN"
"http://www.liferay.com/dtd/liferay-display_5_2_0.dtd>

<display>
    <category name="category.sample">
        <portlet id="maven" />
    </category>
</display>
```

Es la hora de probar el portlet y desplegarlo en el servidor, para ello vamos a configurar en el fichero pom.xml del proyecto una tarea ant que permita que el .war generado en la fase de package se copie en la carpeta deploy del servidor Liferay. Para conseguir esto editamos el fichero pom.xml del proyecto y lo dejamos con el siguiente aspecto:

```
view plain print ?
<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_0_0.xsd'>
<modelVersion>4.0.0</modelVersion>
<groupId>com.autentia.training</groupId>
<artifactId>maven-portlet</artifactId>
<packaging>war</packaging>
<version>1.0-SNAPSHOT</version>
<name>maven-portlet Maven Webapp</name>
<url>http://maven.apache.org</url>
<properties>
    <portlet-hotdeploy>{ruta de la carpeta deploy}</portlet-hotdeploy>
    <portlet-deploy>{ruta carpeta webapps del servidor}</portlet-deploy>
</properties>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.liferay</groupId>
        <artifactId>portlet</artifactId>
        <version>1.0</version>
        <scope>provided</scope>
        </dependency>
    <dependency>
    <groupId>com.liferay</groupId>
        <artifactId>portal-kernel</artifactId>
        <version>1.0</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
<build>
    <finalName>maven-portlet</finalName>
    <plugins>
        <plugin>
            <artifactId>maven-antrun-plugin</artifactId>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>Borrando a ${portlet-deploy}/maven-portle</echo>
                            <delete dir="${portlet-deploy}/maven-portlet' />
                            <echo>Copiando a ${portlet-hotdeploy}</echo>
                            <copy todir="${portlet-hotdeploy}" file="${project.build.directory}/${project.build.finalName}.war></copy>
                        </tasks>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
</project>
```

Lamentablemente a día de hoy no tenemos las librerías de Liferay en ningún repositorio oficial por lo que tendremos que instalar a mano, como mínimo para este ejemplo, las dependencias portlet y portal-kernel cuyos .jar se encuentran en en el directorio LIFERAY_TOMCAT_HOME/libs/ext.

Para hacer esto debemos ejecutar lo siguiente en un terminal del sistema para cada una de las librerías:

```
view plain print ?
mvn install:install-file -Dfile=ruta_absoluta_jar -DgroupId=com.liferay -DartifactId=portal-kernel o portlet -Dversio1.0 -Dpackaging=jar
```

Con esta configuración estamos consiguiendo que cuando ejecutemos mvn package del proyecto, esto genere un archivo .war y haga un deploy limpio en el servidor que le indiquemos en las propiedades.

Para probar el despliegue podemos editar el fichero view.jsp añadiendo un mensaje del tipo: ¡Este portlet esta hecho con Maven!.

Una vez realizado el cambio, guardamos y ejecutamos en un terminal dentro del directorio raíz del proyecto:

```
view plain print ?
mvn package
```

Si el servidor Liferay ya esta arrancado veremos en las trazas de log que se produce el despliegue en caliente, sino después de ejecutar el comando, arrancamos el servidor.

Y este tiene que ser el resultado final del portlet:

2009-07-02
Axis2: Invocación de Servicios Web usando distintos MEP

2009-07-02
Instalación OpenOffice

2009-07-02
Juegos 3D en Java: Blender y JMonkeyEngine

2009-06-20
StAX (Xml Pull Parser): Streaming API para XML

2009-06-15
Configuración de la desconexión de usuarios con ICEFaces

2009-06-10
LWUIT: Una librería gráfica tipo AWT o Swing para J2ME

Últimas ofertas de empleo

2009-07-31
T. Información - Operador (dia / noche) - BARCELONA.

2009-06-25
Atención a cliente - Call Center - BARCELONA.

2009-06-19
Otras - Ingenieria (minas, puentes y puertos) - VALENCIA.

2009-06-17
Comercial - Ventas - ALICANTE.

2009-06-03
Comercial - Ventas - VIZCAYA.

Anuncios Google

4. Conclusiones.

Cómo veis no tenemos porque renunciar a todas las ventajas que nos ofrece la forma de trabajo con Maven sólo porque tengamos que desarrollar portlets en Liferay. Sólo hay que echarle un poco de inventiva y adaptar una cosa con la otra.

Saludos.

¿Qué te ha parecido el tutorial? Déjanos saber tu opinión y ¡vota!

Muy malo Malo Regular Bueno Muy bueno

Votar

(Sólo para usuarios registrados)

» Registrate y accede a esta y otras ventajas «

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

Creative Commons License Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

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

J2EE, EJBs, Struts...

Tutoriales recomendadosxx

Nombre **Resumen** **Fecha** **Visitas** **Valoración** **Votos** **Pdf**

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.