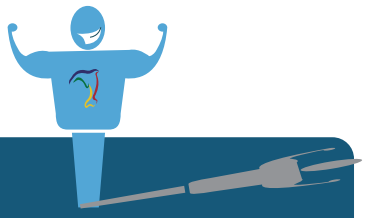
autentia

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...

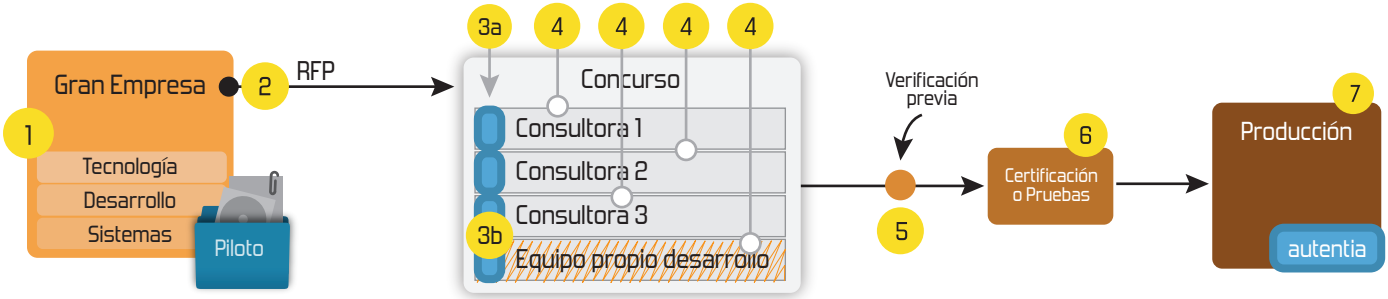
1. Desarrollo de componentes y proyectos a medida



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

Tutorial desarrollado por: Alejandro Perez García 2003-2005, nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Si te gusta lo que ves, **puedes contratarme** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

Contacta: alejandropg@autentia.com.

Como informar a Debian GNU/Linux de nuestra Máquina Virtual Java

Creación: 11-08-2005

Índice de contenidos

1. Introducción

Hoy en día es muy normal utilizar alguna aplicación escrita en Java. Para poder ejecutar este tipo de aplicaciones es necesario tener instalada una Máquina Virtual de Java (a partir de ahora la nombraremos como JVM) en nuestro sistema.

En Debian tenemos varias opciones para hacer esto. De hecho podemos encontrar entre los paquetes oficiales varias JVM, como: gij (intérprete de byte code), ikvm (JVM para .NET), jamvm, kaffe, ...

En mi caso he instalado una JVM que no está entre los paquetes de Debian. En concreto he instalado la JVM de Sun Microsystems. Este tipo de JVM no las encontraremos en los paquetes oficiales de Debian por problemas de licencias, ya que no son compatibles con el Software Libre.

Al haber instalado una JVM fuera de los paquetes de Debian, mi sistema desconoce que tiene la capacidad para ejecutar aplicaciones Java.

Lo que vamos a ver en este tutorial es precisamente eso, como indicar a nuestro sistema Debian que tenemos una JVM instalada.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Ahtex Signal X-9500M (Centrino 1.6 GHz, 1024 MB RAM, 60 GB HD).
- Sistema Operativo: GNU / Linux, Debian Sid (unstable), Kernel 2.6.12, KDE 3.4
- Máquina Virtual Java: JDK 1.5.0_03 de Sun Microsystems

3. Instalamos los paquetes necesarios

Lo primero que vamos a hacer es instalar java-common y equivs

```
# apt-get -u install java-common equivs
```

Con equivs lo que vamos a hacer es crear unos paquetes que sólo sirven para satisfacer dependencias. Es decir, al instalar estos paquetes el sistema sabrá que "alguien" provee cierta funcionalidad. Vamos a denominar a estos paquetes "paquetes dummy" ya que realmente no tienen contenido.

El paquete java-common lo necesitamos porque los paquetes "dummy" que vamos a creara dependerán de este. Además trae documentación muy interesante.

4. Creamos los paquetes "dummy" de Java

Para esto nos movemos a /usr/share/doc/java-common/dummy-packages (aunque la documentación de java-common recomienda trabajar sobre /var/install/java/pkg).

```
# cd /usr/share/doc/java-common/dummy-packages
```

Ahora hacemos:

```
# equivs-build java2-runtime-dummy.control
# equivs-build java2-compiler-dummy.control
# equivs-build java-virtual-machine-dummy.control
```

Con esto hemos creado tres paquetes listos para instalar. Una vez instalados nuestros sistema sabrá que tenemos un entorno de ejecución y de compilación basado en Java2, por lo que podremos instalar cualquier paquete que necesite estas dependencias, por ejemplo freemind (estupenda aplicación para representar árboles de pensamiento).

5. Instalación de nuestros paquetes "dummy"

Tan sólo tenemos que hacer:

```
# dpkg -i java2-runtime-dummy_1.0_all.deb
# dpkg -i java2-compiler-dummy_1.0_all.deb
# dpkg -i java-virtual-machine-dummy_1.0_all.deb
```

6. Probar que todo ha ido bien

Si no hemos tenido ningún problema, podemos probar si las dependencias están correctamente dadas de alta en nuestro sistema. Para ello vamos a probar a instalar freemind

```
# apt-get -u install freemind
```

Si todo está bien, el paquete se instalará sin ningún problema, y podremos ejecutarlo con:

```
$ freemind
```

Si al ejecutar tenemos algún problema, comprobar que la instalación de vuestra JVM es correcta. Es decir, como mínimo debería estar definida la variable de entorno JAVA_HOME, y los ejecutables de la JVM deberían estar en el PATH.

7. Conclusiones

En este pequeño tutorial hemos visto como informar a nuestro sistema sobre ciertas "habilidades" que están resultas por componentes fuera de la estructura habitual de paquetes. Aunque lo hemos visto para el ejemplo concreto de Java, lo podríais hacer para cualquier dependencia.

Creo que es aconsejable que siempre que instalemos una JVM fuera de los paquetes de Debian, sigamos los pasos de este tutorial para que el sistema tenga constancia de ello.

Por último os recomiendo leer la documentación que viene con el paquete java-common. No es muy extensa y nos explica cual es la

política de Debian frente a Java (además de muchas otras cosas ;).

8. Sobre el autor

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

Dir. Implantación y Rendimiento

Formador en tecnologías J2EE, ADOO, UML

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L.

http://www.autentia.com