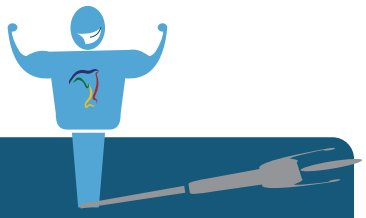
autentia

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...

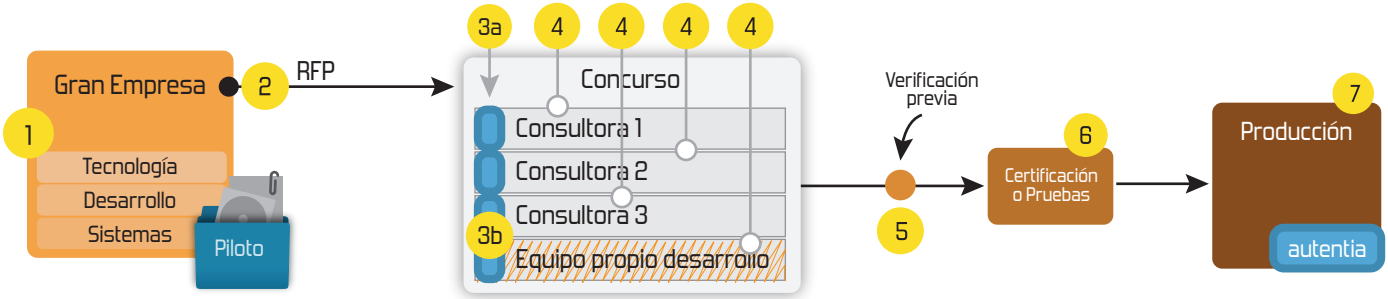
1. Desarrollo de componentes y proyectos a medida



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 2003-2006
Alejandro es Socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

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

Contacta: alejandropg@autentia.com.

autentia real business solutions

Crear un repositorio remoto y como hacer una 'release' con varios proyectos en Maven y Eclipse

Creación: 27-11-2006

Índice de contenidos

1. Introducción

El el anterior tutorial sobre Maven (http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=maven) ya vimos los principios básicos para trabajar con Maven.

En este tutorial vamos a ampliar el punto 13 "Como publicar una nueva 'release' de nuestro proyecto" (http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=maven#mozTocId337503). Donde nos centraremos en como podemos trabajar teniendo varios proyectos relacionados en Maven y en Eclipse, y como podemos crear nuestro propio repositorio remoto para guardar el resultado de nuestras versiones.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Ahtec Signal 259M MXM (Sonoma 2.1 GHz, 2048 MB RAM, 100 GB HD).
- Sistema Operativo: GNU / Linux, Debian Sid (unstable), Kernel 2.6.17, KDE 3.5
- Máquina Virtual Java: JDK 1.5.0_09 de Sun Microsystems
- Maven 2.0.4
- Eclipse 3.2.1
- Subversive Eclipse plugin 1.1.0 M9a

3. Trabajar con más de un proyecto en Maven y en Eclipse

Actualmente Eclipse necesita una estructura plana de proyectos. Con esto queremos decir que todos los proyectos relacionados, sobre los que estamos trabajando, deben de estar dentro del mismo workspace, como proyectos que cuelgan directamente del workspace. Esto se traduce en que no podemos tener un proyecto dentro de otro.

Con Maven podemos tener varios proyectos relacionados sobre los que estamos trabajando, y podemos tener otro proyecto padre (otro `pom.xml`) que se encargue de gestionarlos a todos. Este `pom.xml` padre nos permite hacer una misma operación sobre todos los `pom.xml` hijos, e incluso Maven tendrá en cuenta el orden según las dependencias (por ejemplo si A depende de B, Maven se encargará de compilar primero B y luego A).

En el punto 10 "Un pom.xml para controlarlos a todos" (http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=maven#mozTocId358116) ya vimos como podíamos trabajar con varios proyectos en Maven, y seguir manteniendo esa estructura plana que requiere Eclipse. El quid de la cuestión está en como hacíamos referencia a los módulos 'hijos' en el `pom.xml` 'padre', utilizando '..' para descender un nivel en la jerarquía de directorios:

```
<module>../autentia-negocio</module>
```

Pues resulta que esta estructura de directorios (donde el padre está a la misma altura que los hijos) no está soportada por el plugin "release" de Maven (http://jira.codehaus.org/browse/MRELEASE-6?rc=1), por lo que tendremos problemas si pretendemos hacer una 'release' conjunta de todos los proyectos 'hijos'.

> Nota: En general no es un problema de este plugin, sino de la gestión de la herencia en Maven.

El plugin "release" requiere que en la estructura de directorios el proyecto padre sea, físicamente, el directorio padre de los proyectos hijos. Pero esta estructura no casa con lo que hemos dicho antes sobre el Eclipse, ¿como resolverlo?

Bueno, en este tutorial vamos a ver lo que a mi me ha parecido la forma más cómoda de resolver este problemilla, pero no digo que sea la única, e invito a que cada uno busque la forma de trabajar que le resulte más cómoda.

Supondremos que la estructura de directorios es la siguiente:

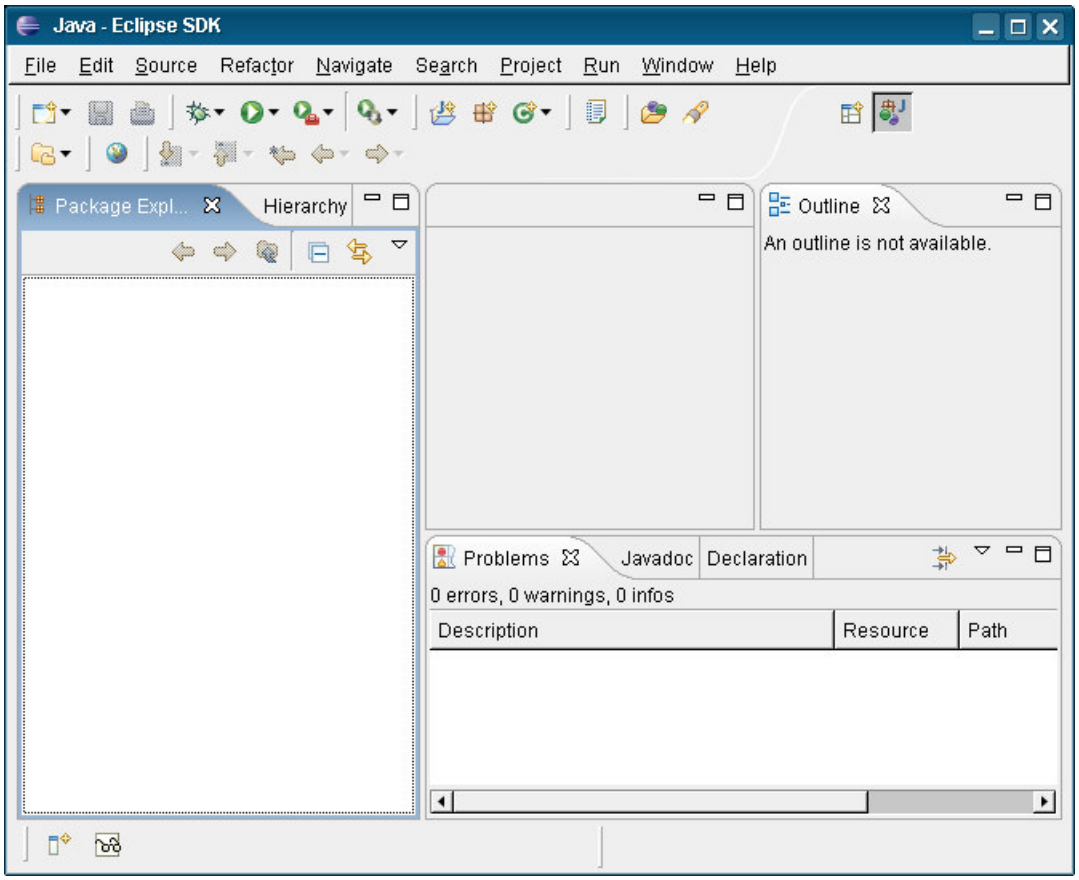
```
autentia-parent
|-- pom.xml
|-- autentia-negocio
|   `-- pom.xml
`-- autentia-web
    `-- pom.xml
```

Esta es la estructura necesaria para que Maven y sus plugins funcionen correctamente, y por lo tanto, esta será la estructura que encontraremos en el repositorio.

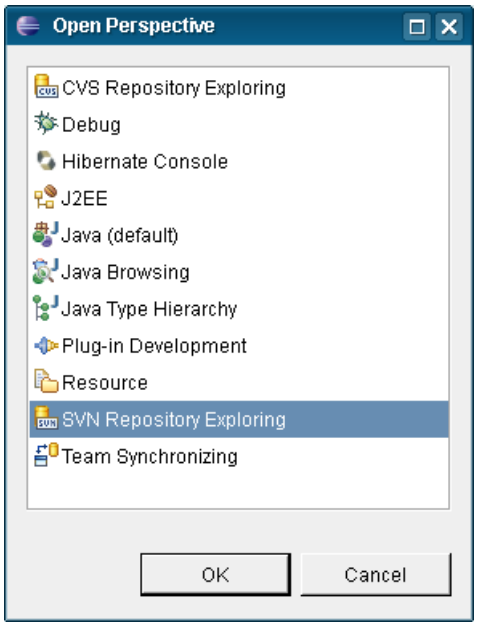
> Nota: Acordaros que siguiendo esta estructura de directorios, en el `pom.xml` del padre se hará referencia a los hijos con: `<module>autentia-negocio</module>`

3.1. Dar de alta el repositorio en Eclipse

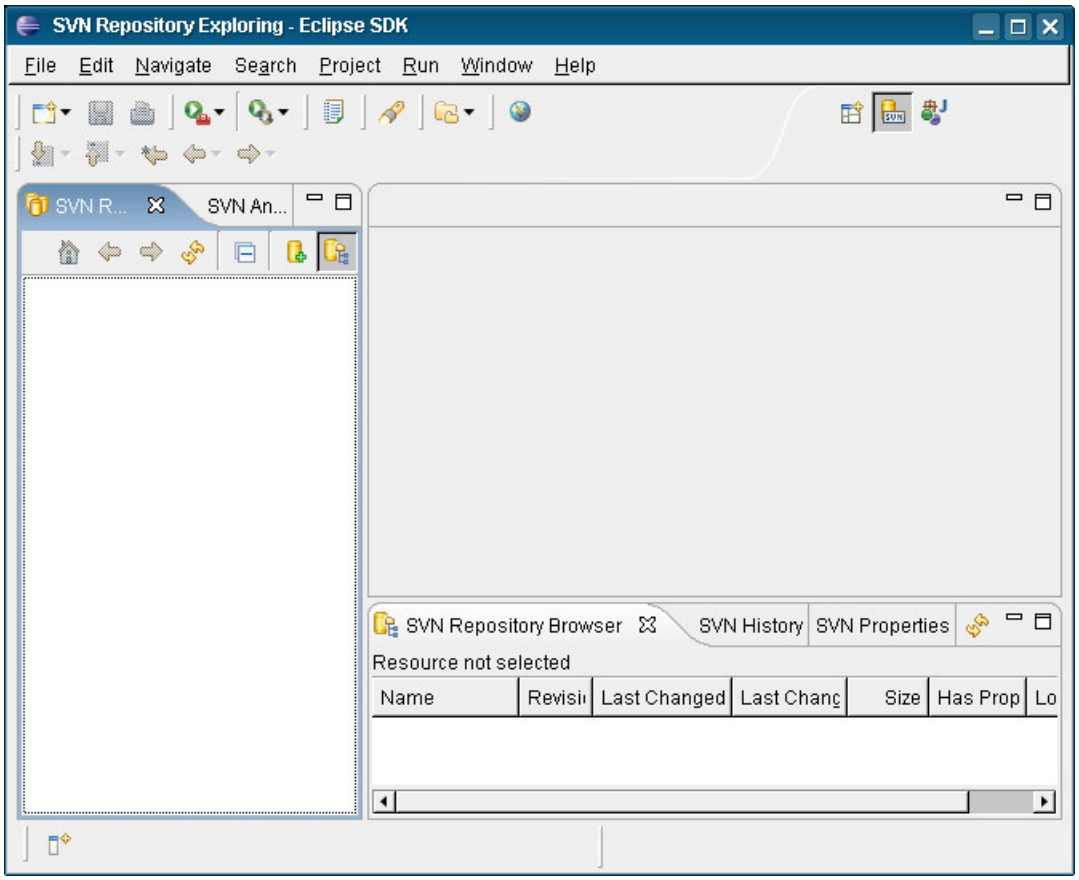
Al principio tendremos nuestros workspace vacío.



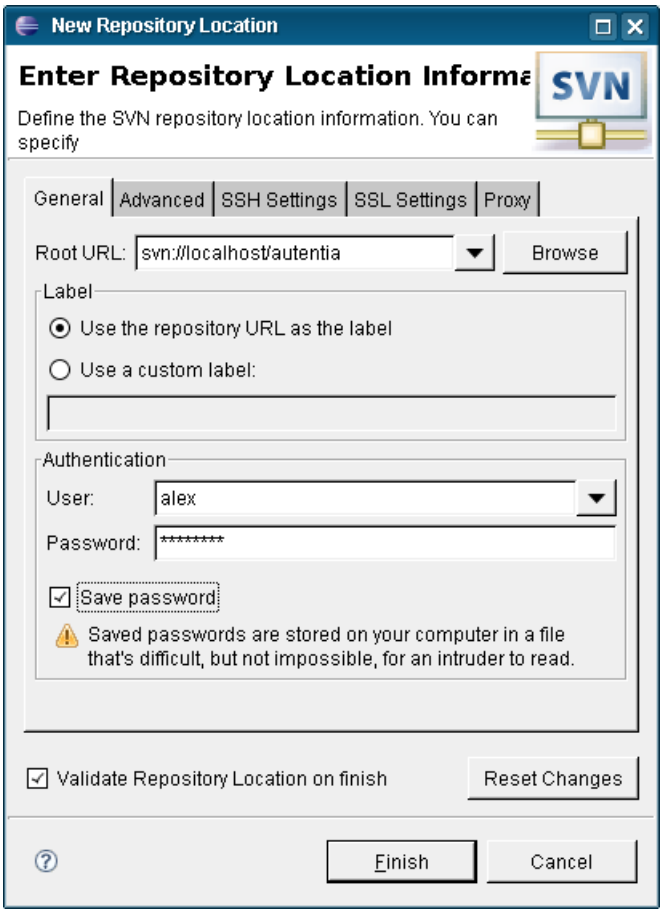
Lo primero que vamos a hacer es dar de alta el repositorio en el Eclipse: Window --> Open Perspective --> Other ...



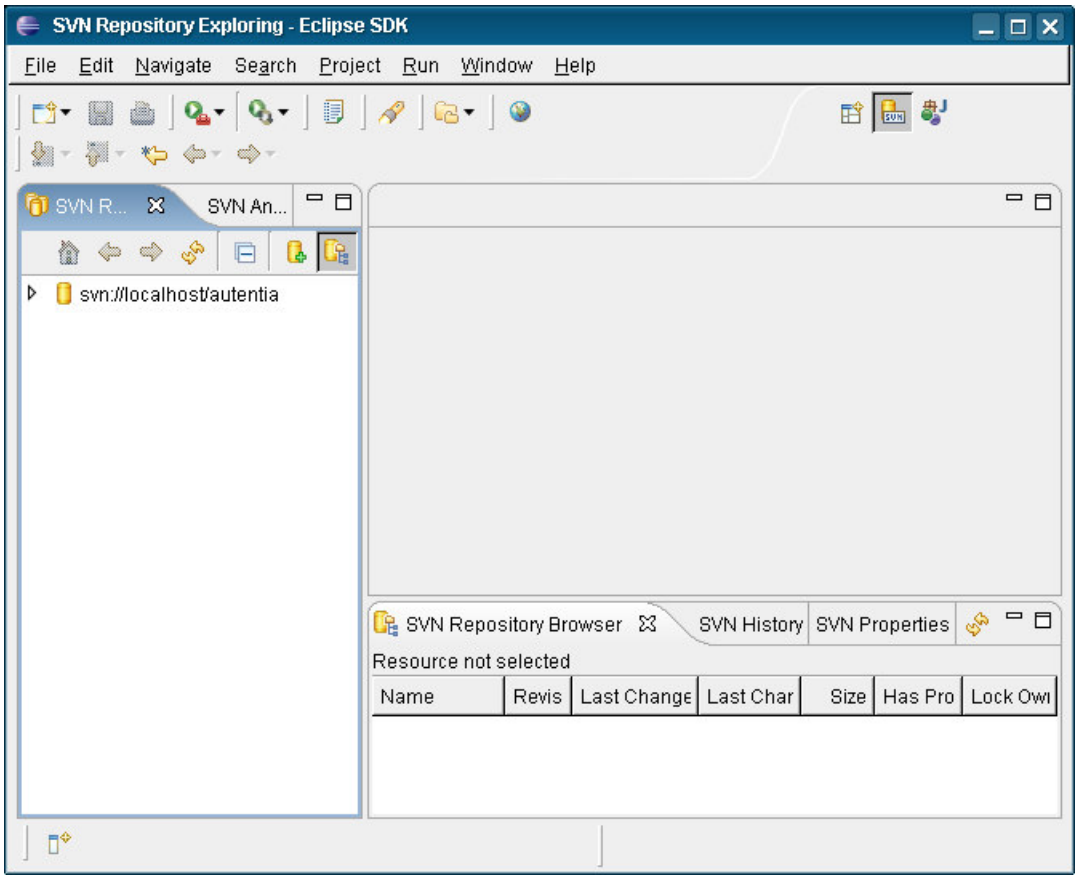
Seleccionamos "SVN Repository Exploring" y pulsamos "OK"



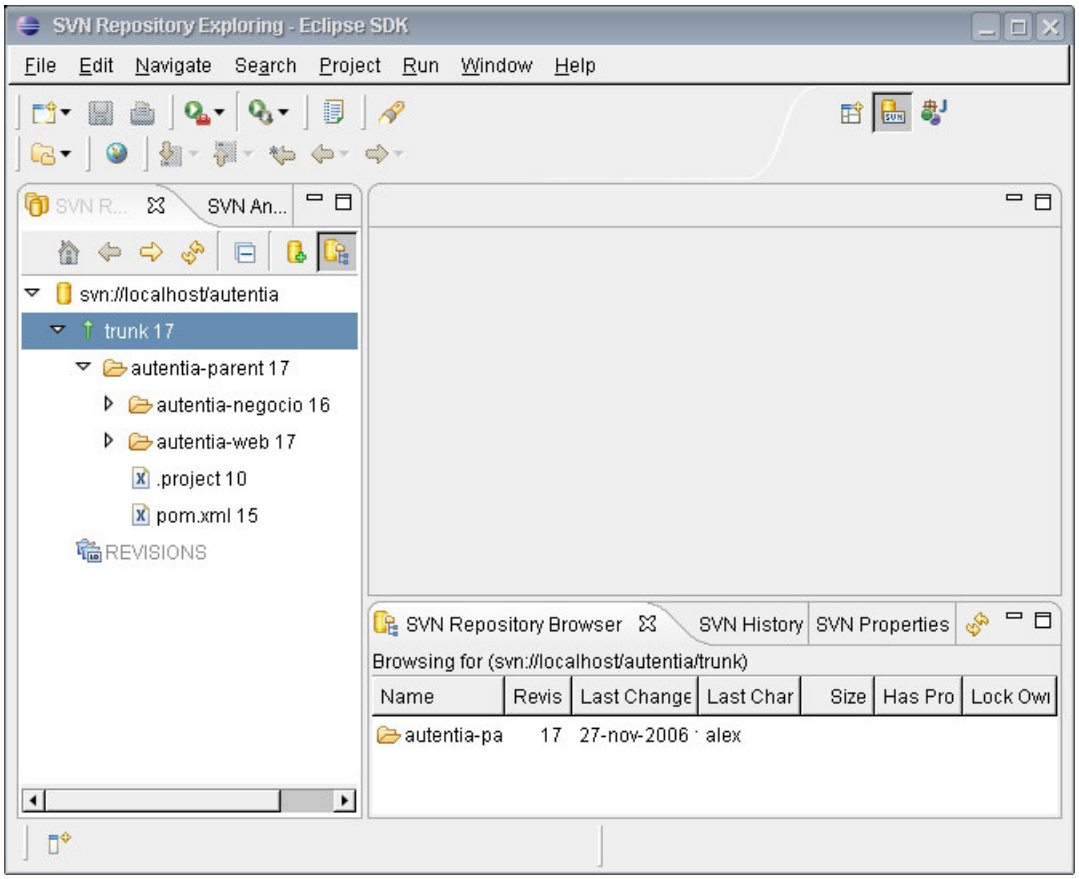
File --> New --> Repository Location



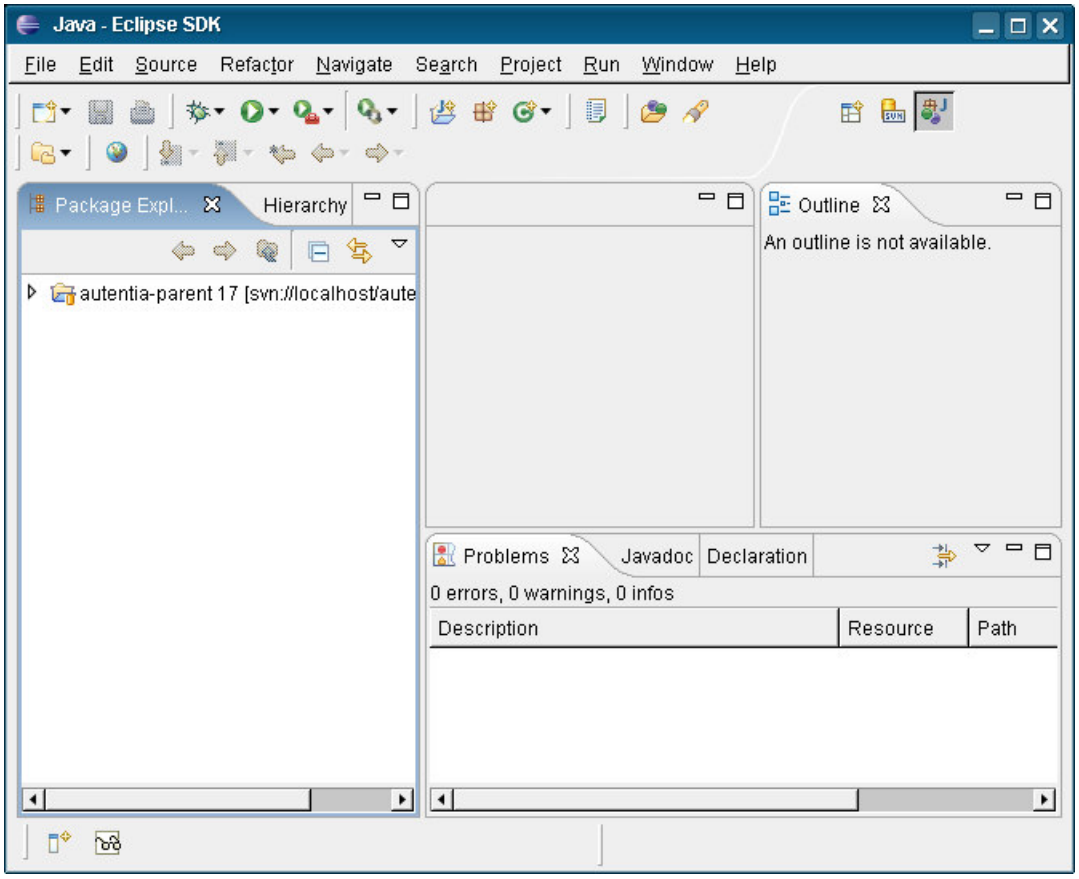
Introducimos los datos para acceder a nuestro repositorio de Subversion y pulsamos "Finish".



Si vamos desplegando las carpetas del repositorio (pinchamos sobre el triángulo a la izquierda de "svn://localhost/autentia"), veremos como aparece la estructura de directorios que hemos comentado antes, donde "autentia-negocio" y "autentia-web" cuelgan de "autentia-parent".

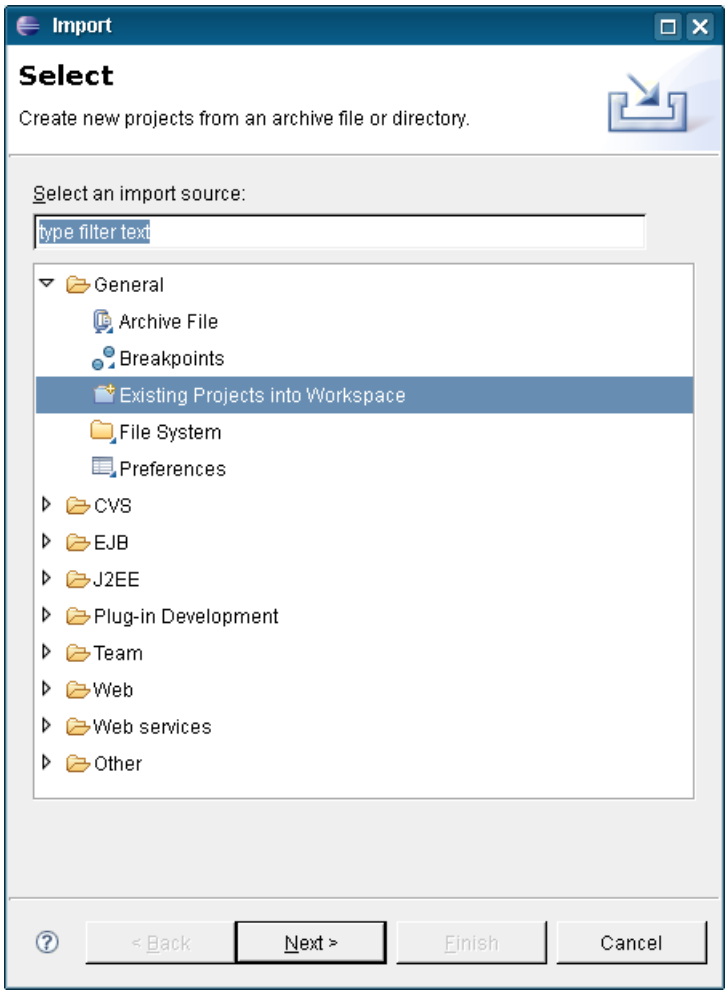


Nos bajamos el proyecto "autentia-parent": Botón derecho sobre "autentia-parent" --> Check Out. Esto nos vaja el proyecto "autentia-parent", y como podréis imaginar, también "autentia-negocio" y "autentia-web", ya que están dentro de aquel. El problema es que estos dos últimos no los va a reconocer el Eclipse, puesto que están dentro del primero. Lo podemos comprobar si saltamos nuevamente a la perspectiva de Java:

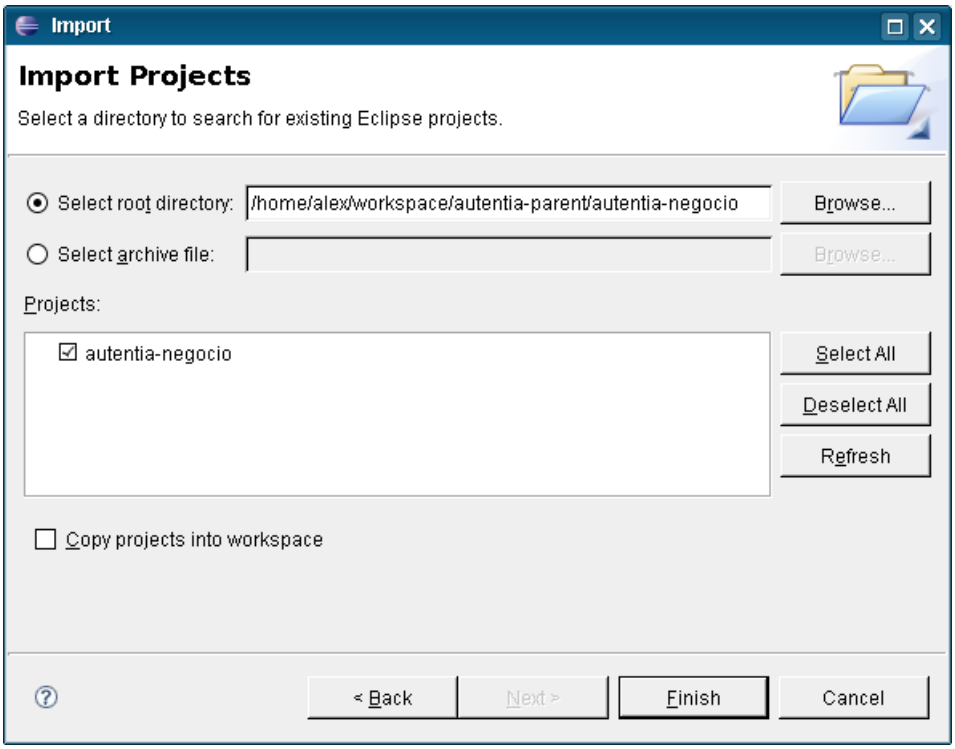


3.2. Dar de alta en el workspace a los proyectos hijos

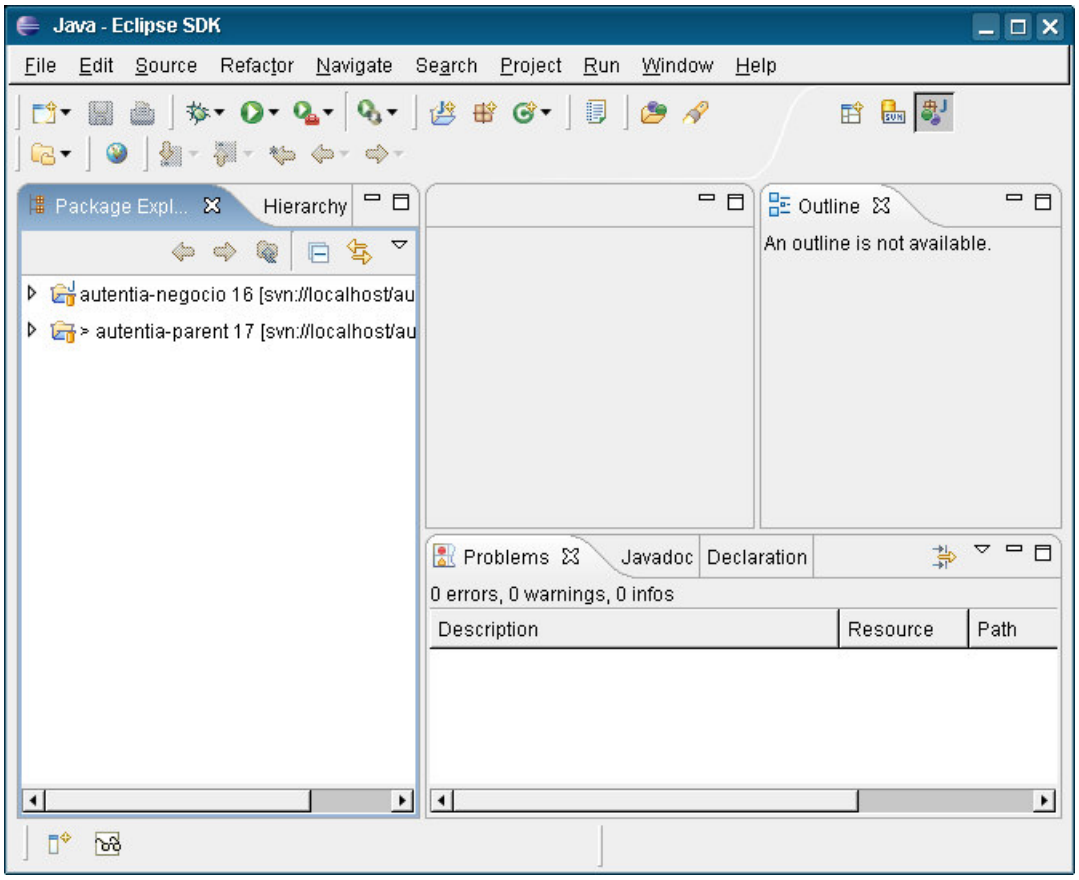
Ahora vamos a importar dentro del workspace los dos proyectos que están dentro de "autentia-parent". De esta forma tendremos los tres proyectos en el workspace y podremos trabajar con cualquiera de ellos. Para ello: File --> Import...



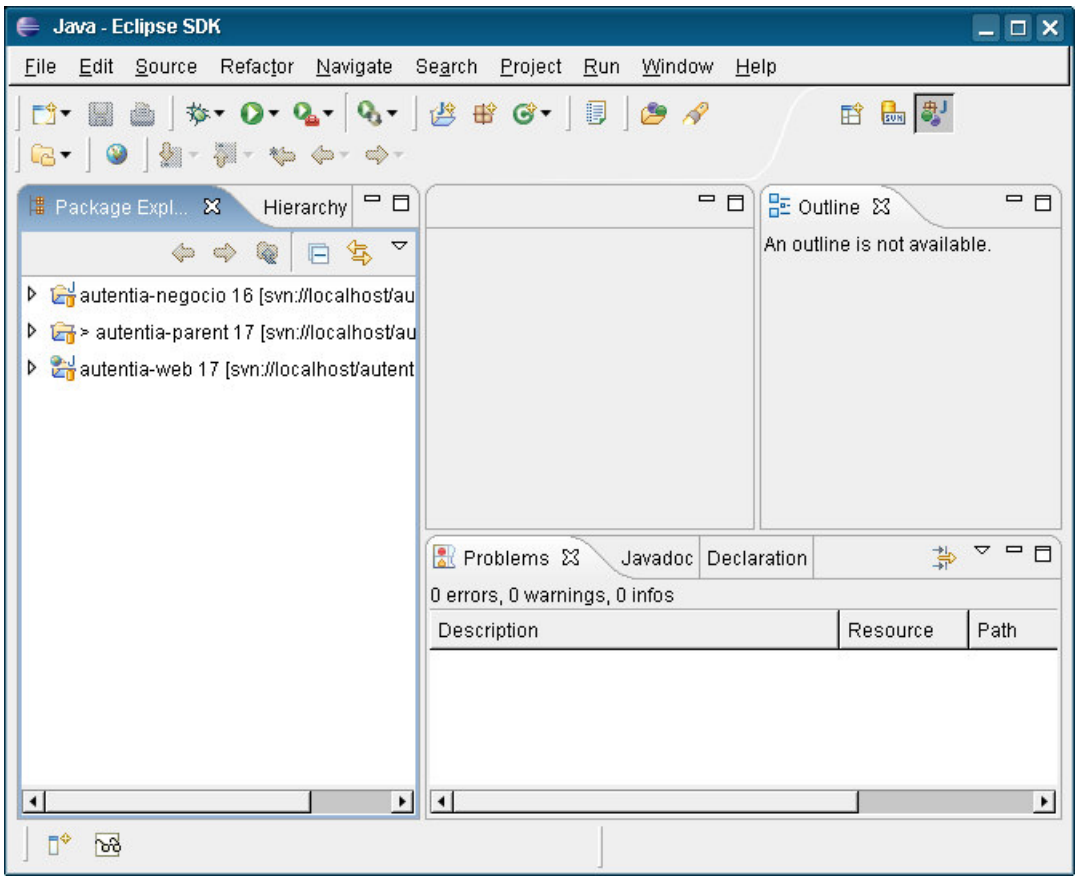
Pinchamos sobre "Existin Projects into Workspace", y pulsamos "Next >".



Introducimos la ruta a uno de los proyectos, y vemos como nos aparece el proyecto en el área de "Projects:". Pulsamos "Finish".



Vemos como ya tenemos el proyecto en el workspace. Ahora repetimos la misma operación de importación para el proyecto "autentia-web".



Ahora ya podemos trabajar sobre los proyectos con normalidad. Aunque el proceso no se puede decir que es 'intuitivo', nos permite trabajar con Maven y con Eclipse sin perder la potencia de ninguno de los dos.

Dentro de este pequeño "galimatías" una ventaja es que si tenemos muchos proyectos hijos y queremos sincronizarlos con el

repositorio, basta con sincronizar el padre, y no tenemos que ir uno por uno (o seleccionarlos todos).

Otra ventaja es la puramente organizativa. Podemos tener dentro de un mismo repositorio (por ejemplo un repositorio para cada cliente), más de un proyecto padre, de donde cuelgan sus correspondientes hijos (por ejemplo, cada proyecto padre puede representa una aplicación diferente), así no se nos mezclan los proyectos, quedando claro a que aplicación pertenece cada proyecto.

4. Crear nuestro propio repositorio remoto

En Maven trabajamos con un repositorio local, situado en nuestra máquina, y con un repositorio remoto. Habitualmente el repositorio remoto está en www.ibiblio.org, pero hay muchos componentes que no se encuentran en este repositorio y que nosotros queremos usar en nuestros proyectos, por ejemplo el jta de Sun. Para esto nos podemos hacer nuestro propio repositorio donde colgaremos todas las dependencias que no podamos encontrar en ibilbio u otro repositorio disponible en Internet.

Otro motivo para tener nuestro propio repositorio remoto es poder publicar en él nuestros propios proyectos, de forma que estén disponibles para el resto de nuestra organización. El plugin "release" de Maven es el que se encargará de hacer esta publicación.

4.1. Como subir nuestros proyectos al repositorio remoto

En primer lugar vamos a configurar el `pom.xml` del padre para indicarle el repositorio remoto donde debe dejar el proyecto al hacer "deploy" o al usar el plugin "release":

```
<project>
    ...
    <distributionManagement>
        <repository>
            <id>autentia-repository</id>
            <name>Autentia Repository</name>
            <url>scp://servidorDeAutentia/var/maven/repository</url>
        </repository>
        <snapshotRepository>
            <id>autentia-snapshotRepository</id>
            <name>Autentia Snapshot Repository</name>
            <url>scp://servidorDeAutentia/var/maven/snapshotRepository</url>
        </snapshotRepository>
    </distributionManagement>
    ...
<project>
```

Como se puede ver en el ejemplo estamos configurando dos repositorios, uno para las versiones normales y otro para las SNAPSHOT (por ejemplo si estamos haciendo compilaciones nocturnas). Podríamos tenerlo todo sobre el mismo repositorio (definimos sólo el elemento `<repository>`) pero de esta forma nos queda mejor organizado.

Vemos también como le damos un `<id>` a cada repositorio. Este identificador a de ser único.

Y lo más importante, el elemento `<url>`. Con este elemento indicamos donde se tiene que copiar el resultado de empaquetar el paquete. En esta url lo primero que indicamos es el protocolo que ha de usar para hacer la copia, podría ser scp, sftp o ftp. Por supuesto el servidor donde reside el repositorio remoto ha de soportar alguno de estos protocolos. En nuestro ejemplo hemos elegido scp. Luego vemos como estamos indicando la ruta donde se encuentra el repositorio, en nuestro ejemplo "/var/maven/repository".

Ahora sólo nos queda indicar cual será el usuario y la clave que se debe utilizar para comunicarse con el servidor a través de los protocolos scp, sftp o ftp. Para ello vamos a modificar el fichero `settings.xml` que se encuentra en el directorio `.m2` dentro del home del usuario:

```
<settings>
    ...
    <servers>
        <server>
            <id>autentia-repository</id>
            <username>alex</username>
            <password>myClave</password>
        </server>
        <server>
            <id>autentia-snapshotRepository</id>
            <username>alex</username>
            <password>myClave</password>
        </server>
    </servers>
    ...
</settings>
```

Observar como configuramos el usuario y la clave para cada uno de los repositorios. Nótese también como deben coincidir los elementos `<id>` con los que habíamos definido en el `pom.xml`.

Ahora ya podemos hacer `mvn deploy` o usar el plugin "release", y nuestros proyectos se guardarán en nuestro repositorio remoto.

> Nota: Al trabajar con scp o sftp, es posible que la primera vez nos pida confirmación sobre la firma del servidor.
> También es importante que el usuario que estamos usando tenga permiso de escritura sobre el directorio del repositorio.

4.2. Como subir un jar de un 3º al repositorio remoto

Para subir un jar de un tercero a nuestro repositorio remoto basta con ejecutar:

```
mvn deploy:deploy-file -DgroupId=<group-id> \
    -DartifactId=<artifact-id> \
    -Dversion=<version> \
    -Dpackaging=<type-of-packaging> \
    -Dfile=<path-to-file> \
    -DrepositoryId=<id-to-map-on-server-section-of-settings.xml> \
    -Durl=<url-of-the-repositor-to-deploy>
```

Por ejemplo, para subir el jta de Sun, haremos:

```
mvn deploy:deploy-file -DgroupId=javax.transaction \
    -DartifactId=jta \
    -Dversion=1.0.1B \
    -Dpackaging=jar \
    -Dfile=jta-1.0.1B.jar \
    -DrepositoryId=autentia-repository \
    -Durl=scp://romulo/var/autentia/maven/repository
```

De esta forma el jta de Sun estará disponible en nuestro repositorio remoto para ser usado por cualquier proyecto.

5. Como bajar proyectos del repositorio remoto

Ahora lo normal es que queramos usar los proyectos que hemos subido a nuestro repositorio remoto. Para poder usar esos proyectos tenemos que dar de alta los repositorios remotos que hemos creado en el punto anterior. Para ello vamos a modificar el `pom.xml` del proyecto padre (lo pondremos delante del elemento `<distributionManagement>` que habíamos escrito antes):

```
<project>
    ...
    <repositories>
        <repository>
            <id>autentia-repository</id>
            <name>Autentia Repository</name>
            <url>http://elServidorWebDeAutentia/autentia-repository</url>
        </repository>
        <repository>
            <id>autentia-snapshotRepository</id>
            <name>Autentia Snapshot Repository</name>
            <url>http://elServidorWebDeAutentia/autentia-snapshotRepository</url>
        </repository>
    </repositories>
    ...
</project>
```

Podemos ver como definimos la URL donde se encuentra publicado el repositorio (ojo, tenemos que tener un servidor web que nos de visibilidad sobre los repositorios).

> Nota: La definición de repositorios también se puede hacer en el `settings.xml`. Bastaría con mover todo el elemento `<repositories>` al `settings.xml` dentro de un elemento `<profile>`.

6. Conclusiones

Nuevamente insistimos desde Autentia (http://www.autentia.com) en la necesidad de usar estándares y de no estar reinventando continuamente la rueda. En este sentido Maven es un gran aliado en todo el proceso de compilación, pruebas, empaquetado, versionado, gestión de repositorio, ...

Y aunque hay que reconocer que actualmente la documentación no es todo lo completa que nos gustaría, el esfuerzo compensa con creces. Y, además, para eso escribimos estos tutoriales ;)

7. Sobre el autor

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

Socio fundador de Autentia (Formación, Consultoría, Desarrollo de sistemas transaccionales)

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

http://www.autentia.com

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
Subversive, cliente de Subversion para Eclipse	En este tutorial os enseñamos a utilizar este plugin de eclipse que permite trabajar con repositios de Subversion
Manejo de Repositorios CVS desde Eclipse	En este tutorial os enseñamos a manejar el repositorio CVS desde la plataforma Eclipse
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
Optimizando código Java con Eclipse Test & Performance Tools Platform	En este tutorial vamos a aprender como usar Eclipse Test & Performance Tools Platform (TPTP), que nos permite analizar nuestro código
Framework desarrollo eclipse	Aquí os mostramos algunas de las características de Eclipse
Callisto, nunca antes resultó tan fácil desarrollar con Eclipse	En este tutorial os enseñamos a instalar y utilizar Callipso: una aplicación que permite instalar de manera fácil y cómoda plugins y sus dependencias en Eclipse
Repositorio CVS en Windows	Os mostramos como montar un servidor para el control de versiones CVS en Windows asi como acceder a él a través de WinCVS
Soporte XML en Eclipse con X-MEN	Alejandro Perez nos enseña como potenciar el entorno eclipse para facilitarnos el trabajo con ficheros xml, gracias al pluggin X-MEN
Optimización Java con Eclipse Profiler Plugin	Alejandro Pérez nos enseña como analizar el rendimiento de nuestras aplicaciones con Eclipse Profiler Plugin.
Java Server Faces con Eclipse	Este tutorial es un completo estudio de JSF, una moderna tecnología que nos permite desarrollar aplicaciones empresariales robustas.

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