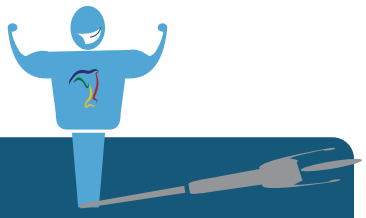
autentia

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...

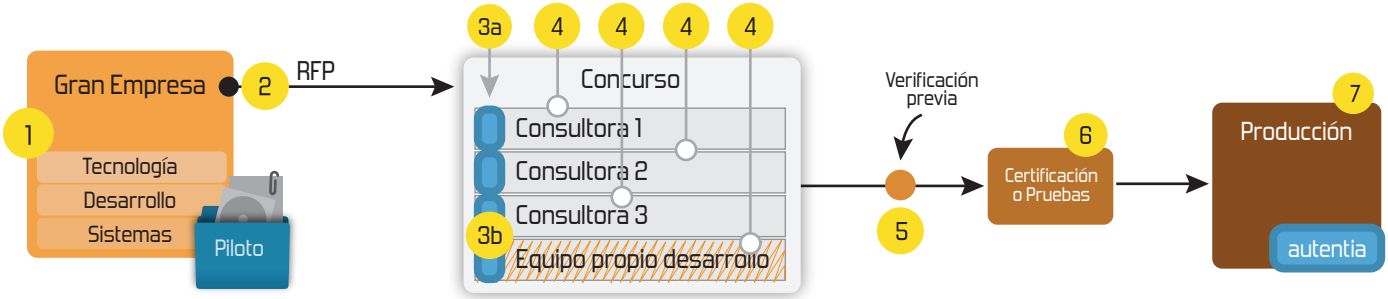
1. Desarrollo de componentes y proyectos a medida



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 2003-2004.

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.

Descargar este documento en formato PDF eclipseProfiler.pdf

Como analizar el rendimiento de nuestras aplicaciones con Eclipse Profiler Plugin

1. Introducción

En este tutorial vamos a ver una poderosa herramienta para analizar el rendimiento de nuestras aplicaciones Java.

Esta herramienta es "Eclipse Profiler Plugin", un plugin de Eclipse que nos va a permitir estudiar a fondo como se distribuye el tiempo de ejecución entre los distintos métodos, clases, ...

A la hora de mejorar el rendimiento de nuestras aplicaciones, es fundamental el uso de una herramienta de este tipo, ya que normalmente el 80% de la perdida de rendimiento está concentrada en un 10% de código. Con este tipo de herramientas vamos a poder identificar cual es ese 10% de código, y no malgastar recursos optimizando código que puede que sólo se ejecute una única vez.

También hay que tener en cuenta que es mucho mejor no centrarnos en la optimización cuando estamos construyendo la aplicación. Es preferible centrarnos en un buen diseño, y en hacer que funcione, y en una etapa posterior y con la ayuda de este tipo de herramientas localizar los puntos críticos (no tiene sentido estar una semana reinventando el algoritmo de ordenación quicksort, para luego descubrir que en la lista vamos a tener un máximo de 5 elementos).

2. Instalación

Eclipse Profiler Plugin es un proyecto SourceForge, con licencia CPL. Lo podemos encontrar en:

http://eclipsecolorer.sourceforge.net/index_profiler.html

En esta página podemos encontrar bastante información de como instalarlo, y de las características que nos ofrece.

En primer lugar descargamos el plugin de la página: http://sourceforge.net/projects/eclipsecolorer. En esta página pincharemos sobre el enlace "Download" de la versión 0.5.31 (la última versión al escribir este tutorial, ojo, es para Eclipse 3.0).

Una vez descargado el archivo ru.nlmk.eclipse.plugins.profiler_0.5.31.zip tendremos que descomprimirlo en el directorio de plugin de eclipse. En mi Linux sería:

```
cd /opt/eclipse/plugin
unzip /download/ru.nlmk.eclipse.plugins.profiler_0.5.31.zip
```

Donde /opt/eclipse es el directorio donde tengo instalado Eclipse 3.0, y /download es el directorio donde me he descargado el archivo ru.nlmk.eclipse.plugins.profiler_0.5.31.zip.

Para terminar con la instalación debemos copiar una librería nativa del sistema donde estemos haciendo la instalación (Linux, Windows, ...). Esta librería viene con plugin, así que basta con hacer (en el caso de mi Linux):

```
cd /opt/eclipse3.0/plugins/ru.nlmk.eclipse.plugins.profiler/native
tar -xzf profiler_linux.tgz
cp libProfilerDLL.so /opt/j2sdk1.4.2_04/jre/lib/i386
```

Donde /opt/j2sdk1.4.2_04 es donde tengo instalada la Máquina Virtual Java.

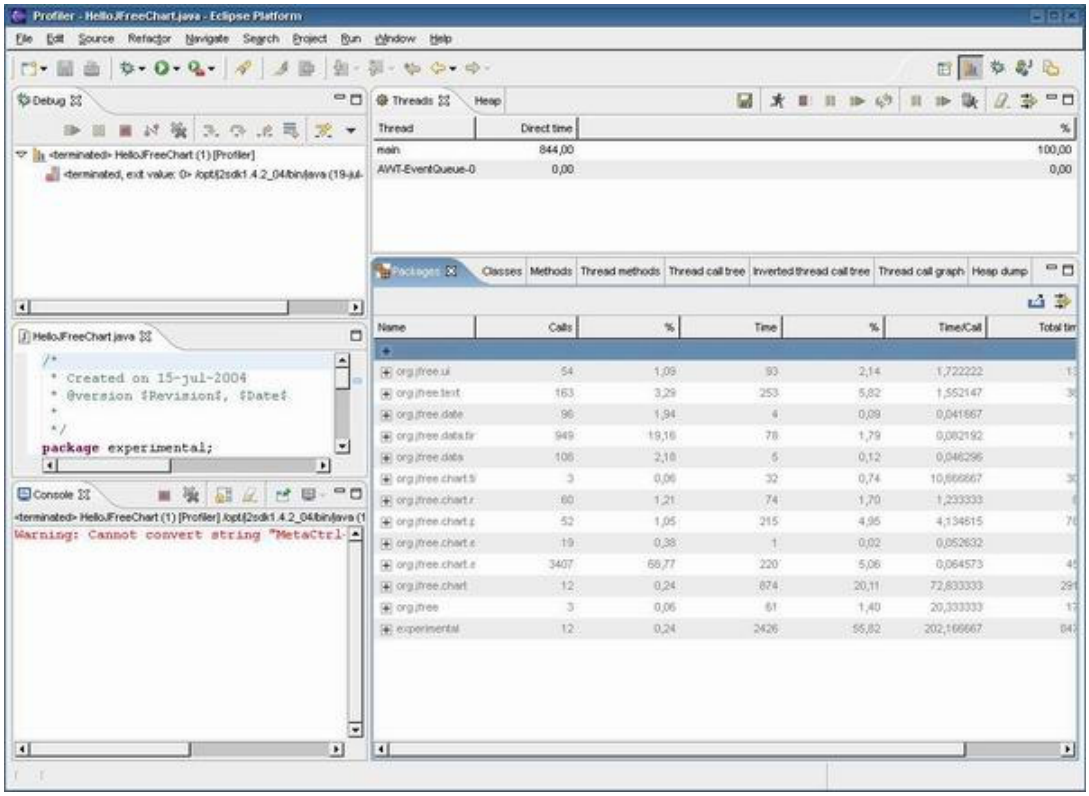
La única consideración es que esta librería esta compilada con gcc 3.2 así que si nuestro sistema no es compatible con esta versió de 3.2 tendremos que compilarla antes de copiarla.

Para que este plugin funcione al 100% es necesario tener instalado GEF (Graphical Editing Framework). Eclipse Profiler Plugin usa este otro plugin para dibujar los grafos de llamadas entre métodos, así que es muy recomendable tenerlo instalado. Lo podemos encontrar en: http://www.eclipse.org/gef/

3. Ejecución

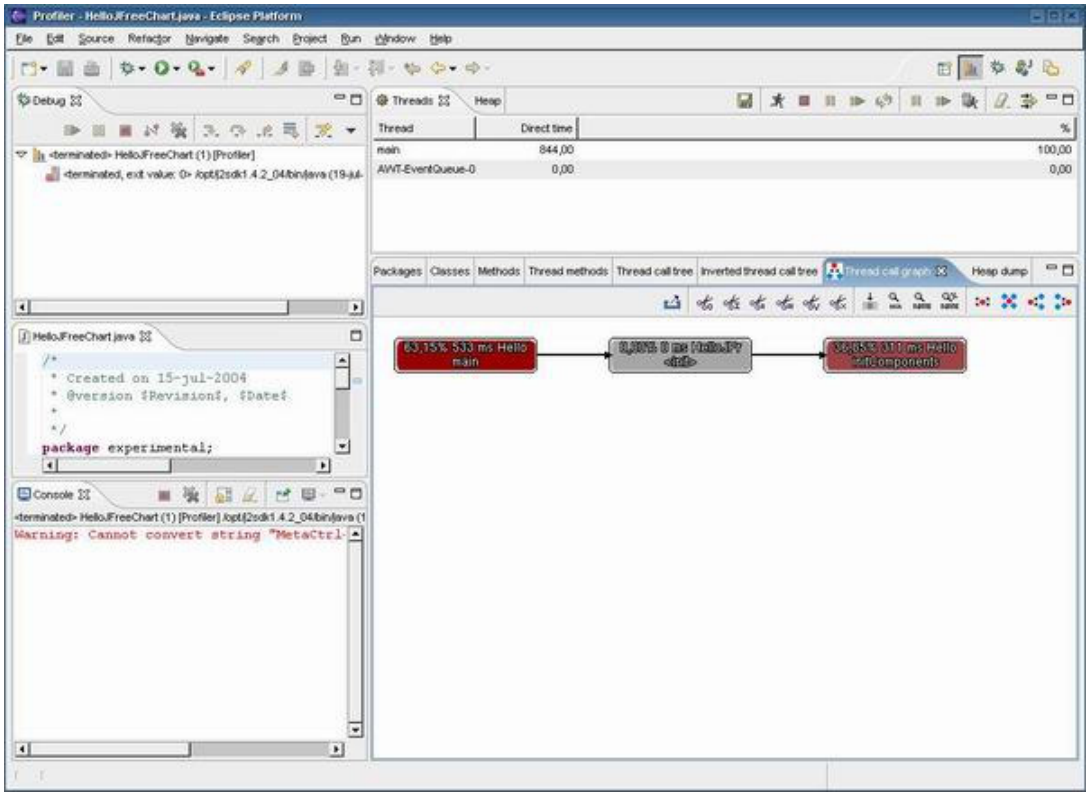
Ahora cuando queramos analizar el rendimiento de nuestra aplicación tendremos que hacer Run --> Run As --> Profiler.

Se abrirá una nueva perspectiva (Profiler perspective), donde veremos el avance de nuestra aplicación. Una vez ha terminado la ejecución es en esta perspectiva donde extraeremos todo el jugo a este plugin.



Como se ve en la imagen anterior podemos visualizar la información por paquete, clase, método, ... En cada una de estas vistas podemos obtener información sobre el número de llamadas, el tiempo por llamada, el tiempo total, ...

Una vista muy interesante es la de "Thread call graph", en esta vista se puede ver el grafo de llamadas entre los distintos método y con un código de colores vemos los métodos que más tiempo han consumido (gris claro poco tiempo, rojo oscuro mucho tiempo Para que se dibuje el grafo es necesario que seleccionemos (en la vista que esta justo encima) el thread que queremos representa



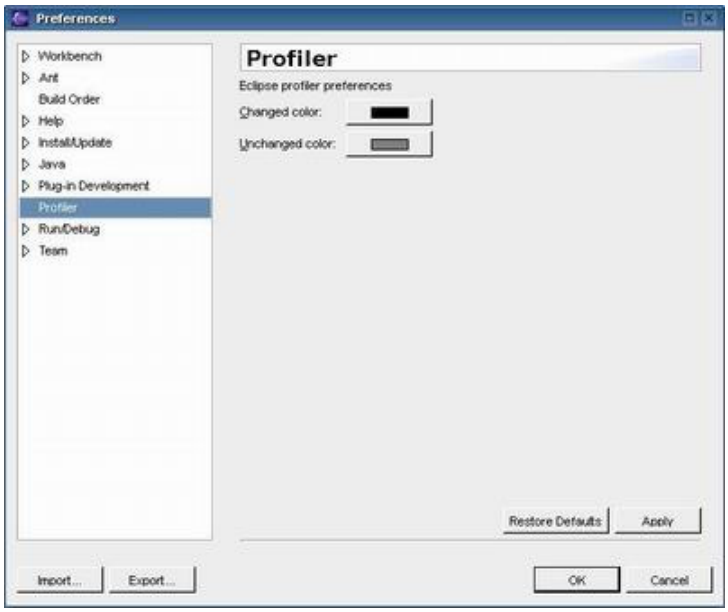
Analizando los datos que pone a nuestra disposición esta perspectiva podemos localizar cual es ese 10% del código que resulta crítico.

Otra cosa que podemos hacer es exportar la información de esta perspectiva a html, de forma que una vez optimizado el código podemos volver a analizar y comparar, comprobando de esta forma que realmente ha mejorado el rendimiento de la aplicación.

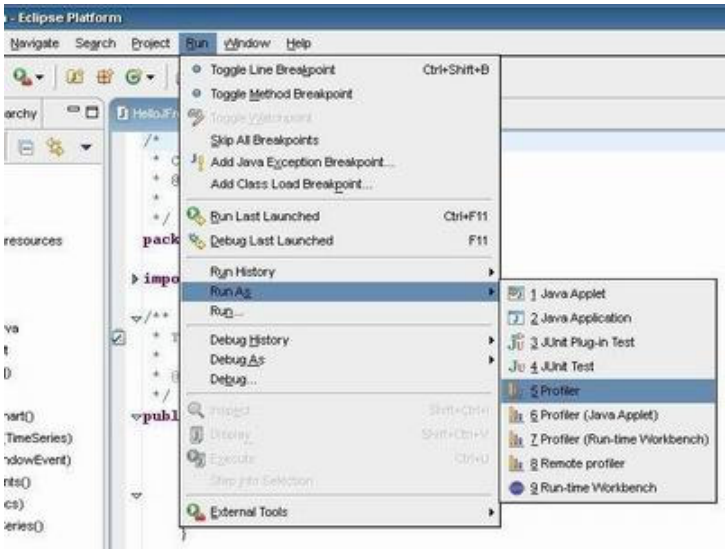
4. Configuración

Una vez terminada la instalación podemos abrir Eclipse.

Si abrimos Window --> Preferences --> Profiler, veremos la pestaña de configuración de Eclipse Profiler Plugin. Como podéis ver e la imagen abajo, es muy sencilla, de hecho lo único que podemos configurar son los colores:

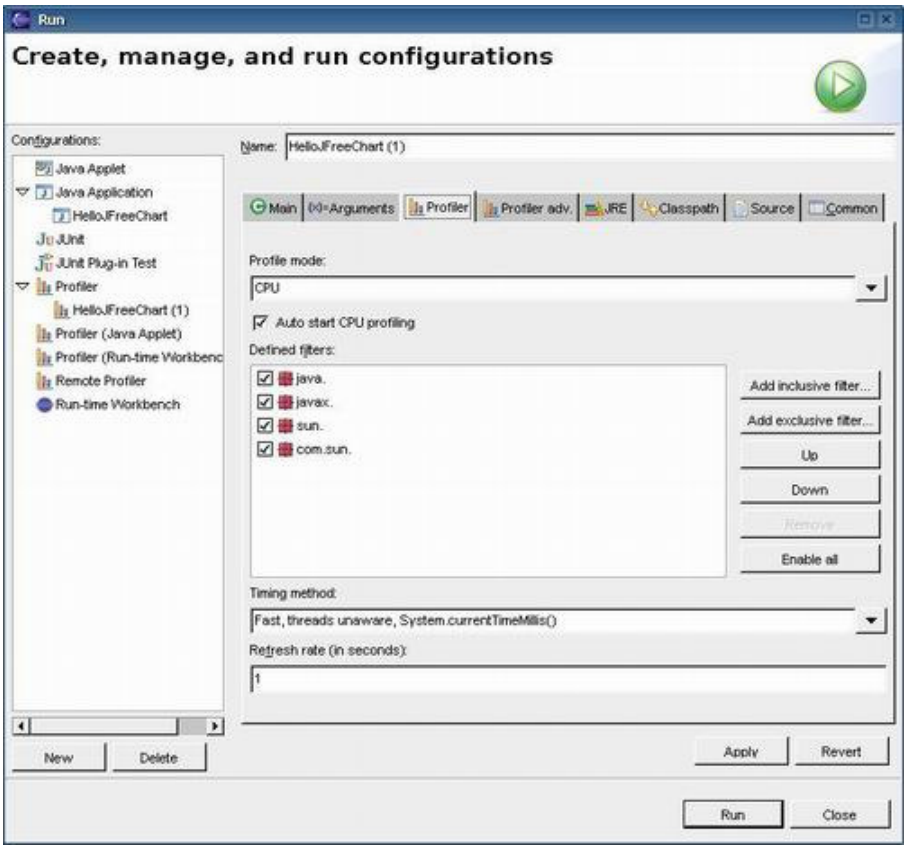


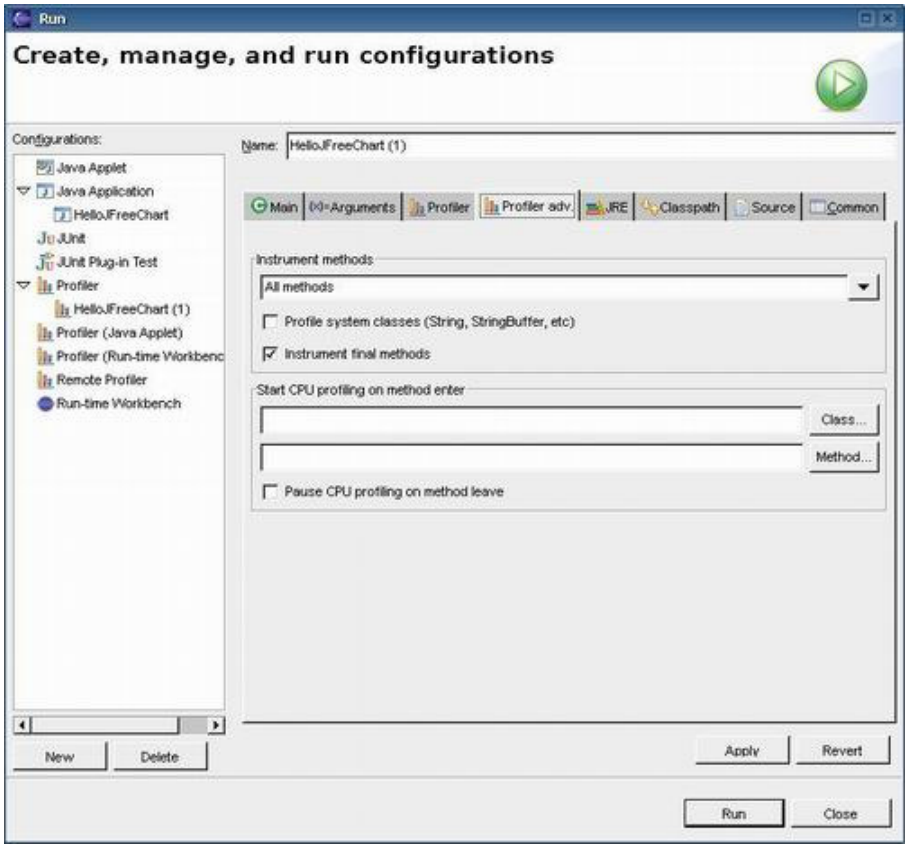
Como se ha comentado en el punto anterior, se han añadido nuevas opciones para ejecutar las aplicaciones desde el menú Run -- Run As:



Si pinchamos sobre Run --> Run... veremos que hay dos pestañas nuevas: Profiler y Profiler adv. Estas dos pestañas nos van a resultar de mucha utilidad para configurar el comportamiento del profiler durante la ejecución. Por ejemplo podemos marcar los paquetes que no queremos que se tengan en cuenta al hacer el profile, como las clases que no hemos desarrollado nosotros, y qu normalmente no modificaremos (en muchos casos ni siquiera tenemos los fuentes de estas clases).

Otra cosa que puede resultar muy útil es indicar cuando debe empezar a contarse los tiempos. Indicando una clase y un método podemos hacer que en vez de contar los tiempos desde que se arranca la aplicación, se haga al llegar a la primera ejecución de es método. Esto puede ser muy útil para saltarnos una primera etapa de inicialización que no queremos contabilizar.





5. Conclusiones

A la hora de escribir aplicaciones y mejorar el rendimiento podemos seguir estos puntos:

- No se debe escribir el código desde un punto de vista de optimización.
- Debemos centrarnos en un buen diseño y lógica de negocio.
- El código debe ser legible para favorecer la mantenibilidad (a la larga, ahorro de recursos).
- Uso de herramientas especializadas, como Eclipse Profiler Plugin.
- Centrar los esfuerzos de optimización en el 10% del código realmente crítico.

Siguiendo estos pequeños consejos, y con las herramientas apropiadas (como Eclipse Profiler Plugin) podemos llegar a un buen equilibrio entre código legible y mantenible, y una aplicación con un excelente rendimiento.

6. Sobre el autor

Alejandro Pérez García

Dir. Implantación y Rendimiento

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L.

http://www.autentia.com

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
Cachear porciones de JSPs	En este tutorial os ensañamos como incrementar increiblemente el rendimiento de vuestro Web basado en tecnología JSP con el FrameWork de cache OSCACHE
Analizar ejecución de programa Java	Os mostramos como investigar el comportamiento de vuestros programas Java, en ejecución, a través del profiling.
Medida del Rendimiento en aplicaciones J2EE	Os mostramos como medir el rendimiento de vuestras aplicaciones Java J2E
Decompilar Java	Os mostramos como recuperar el fuente de vuestro código a partir de los ficheros compilados .class
Programa de Disponibilidad de un Web	Os mostramos como construir una aplicación que os permita monitorizar el funcionamiento de vuestro sistema Web
Rendimiento de aplicaciones Web	En este tutorial veremos una introducción al funcionamiento de la Suite e-Test de Empirix.
AspectJ, Programación con Aspectos	Os mostramos como configurar AspectJ (extensión Java para la programación basada en aspectos) y un pequeño ejemplo para medir la velocidad de una función sin alterar su código.
Aplicaciones con JSPs	Os mostramos como construir una aplicación con JSP que acceda a MySQL
Escritura log con Fichero UDP y JMS	Os mostramos ejemplos para cuantificar el coste de escritura de Logs por pantalla, fichero, UDP y JMS (describiendo como configurar el entorno)
Novedades en Java 1.5	Ya está disponible la versión Beta del J2SDK 1.5. Os mostramos algunas de las nuevas características introducidas en el lenguaje Java: Clases genéricas, enumeraciones, bucles simplificados, etc.