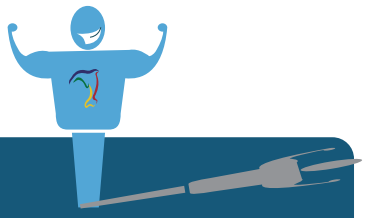
autentia

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...

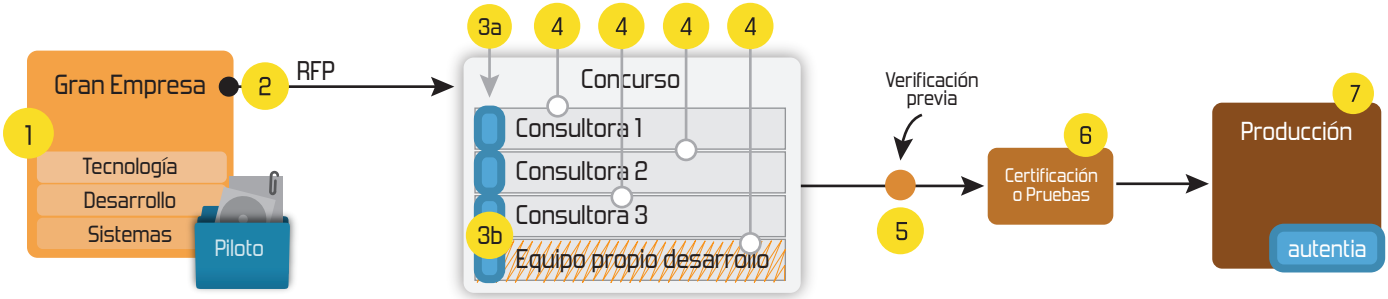
1. Desarrollo de componentes y proyectos a medida



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 | Foros | Tutoriales | Servicios Gratuitos | Contacte



Tutorial desarrollado por:
Roberto Canales Mora 2003-2005
Creador de AdictosAlTrabajo.com y

Director General de Autentia S.L.

Recuerda que me puedes contratar para echarte una mano:

Desarrollo y arquitectura Java/J2EE
Asesoramiento tecnológico Web
Formación / consultoría integrados en tu proyecto

No te cortes y contacta: 655 99 11 72rcanales@autentia.com.

autentia real business solutions

Descargar este documento en formato PDF webserver.pdf

CCS
Centro de Cálculo de Sabadell 1ª empresa española de software
www.ccs.biz

eFax - Fax en su PC
Ahorre tiempo y dinero desde ahora. ¡Pruébelo gratis durante 30 días!
www.efax.com

Global IT Services
Open Source Experts Linux, Apache, Jboss, Asterisk
www.globant.com

Anuncios Goooooogle Anunciarse en este sitio

Creación de nuestro propio servidor Web

Vamos a escribir un pequeño programa para demostrar las capacidades de MultiThread de Java así como exponer de un modo sencillo otras características como son la creación de aplicaciones cliente-servidor con sockets, la escritura en Streams y el control de excepciones, etc ...

Esqueleto básico

Primero vamos a escribir el esqueleto de nuestro programa para no tener que ir repitiendo grandes trozo de código. Este esqueleto podeis observar que esta vacio y no es muy funcional, aunque ya lo iremos rellenando a medida que haga falta.

Lo más importante es el concepto que vamos a tener una función que se llama arranca, dentro de una clase que se habrá encargado de leer los parámetros que le hayan pasado desde la línea de comandos y que disponemos de un método para centralizar los logs.... lo demás ya veremos

```
public class servidorWeb
{
	int puerto = 90;

	final int ERROR = 0;
	final int WARNING = 1;
	final int DEBUG = 2;

    void depura(String mensaje) // los mensajes por defecto serán en modo depuracion
	{
		depura(mensaje,DEBUG);
	}

	// funcion para centralizar los mensajes de depuración
	void depura(String mensaje, int gravedad)
	{
    System.out.println("Mensaje: " + mensaje);
	}

	// punto de entrada a nuestro programa
    public static void main(String [] array)
	{
		servidorWeb instancia = new servidorWeb(array);
		instancia.arranca();
	}

	// constructor que interpreta los parameros pasados
    servidorWeb(String[] param)
	{
		procesaParametros();
```

```
		}

		// parsearemos el fichero de entrada y estableceremos las variables de clase
	boolean procesaParametros()
		{
			return true;
		}

	boolean arranca()
		{
			depura("Arrancamos nuestro servidor",DEBUG);
			return true;
		}

}
```

Bueno, lo primero que tenemos que hacer, es quedarnos a la espera en un puerto, creando un Socket de servidor.

Cuando un navegador nos envie una pertición, la procesamos y mostramos por pantalla

```
boolean arranca()
{
		depura("Arrancamos nuestro servidor",DEBUG);
		try
		{
			ServerSocket s = new ServerSocket(90);
			depura("Quedamos a la espera de conexion");
			Socket entrante = s.accept();

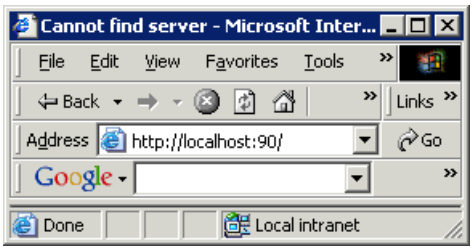

			depura("Procesamos conexion");
			BufferedReader in = new BufferedReader (new InputStreamReader(entrante.getInputStream()));
			String cadena = "";

			while (cadena != null)
			{
				cadena = in.readLine();
				if (cadena != null)
				{
					depura("--" + cadena);
				}
			}

			depura("Hemos terminado");

		}
		catch(Exception e)
		{
			depura("Error en servidor\n" + e.toString());
		}
			return true;
		}
}
```

Si desde un navegador, realizamos una petición a este estilo



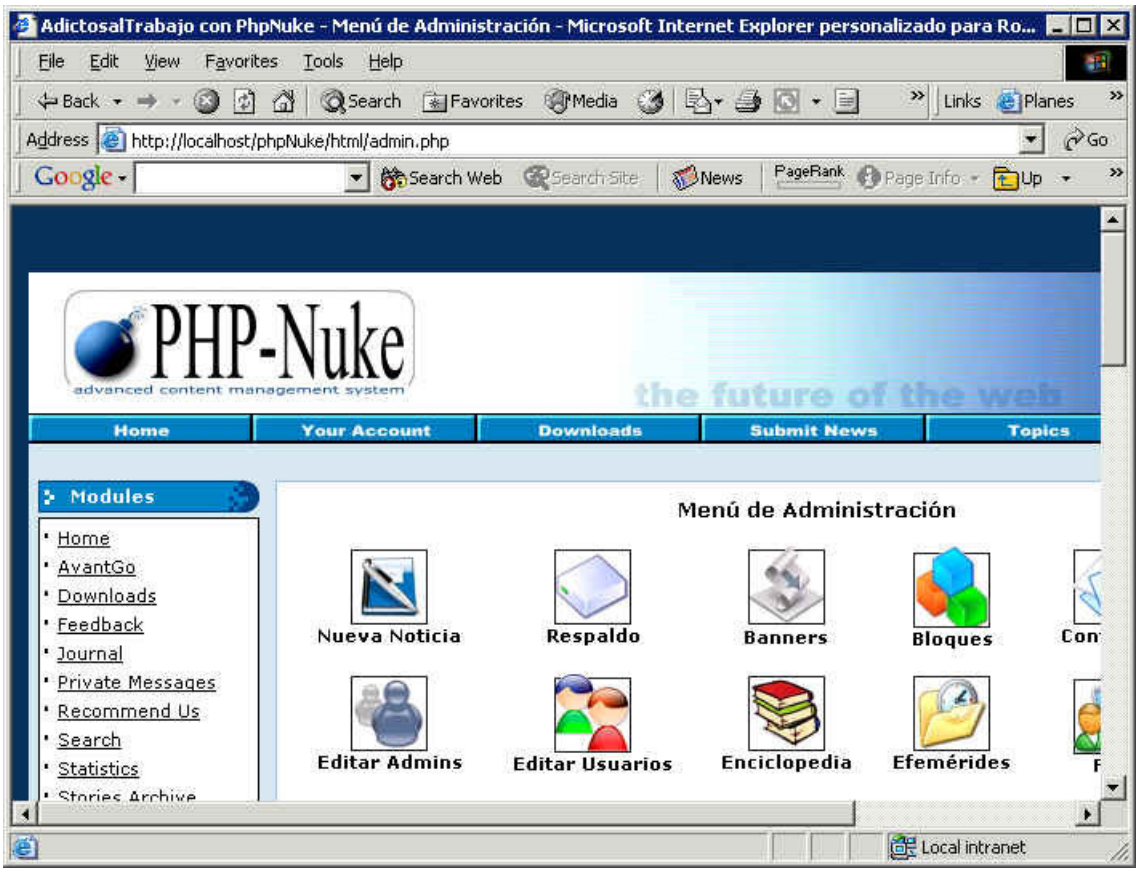
En nuetro log veremos

```
Mensaje: Arrancamos nuestro servidor
Mensaje: Quedamos a la espera de conexion
Mensaje: Procesamos conexion
Mensaje: --GET / HTTP/1.0
Mensaje: --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Mensaje: --Accept-Language: es
Mensaje: --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Mensaje: --Host: localhost:90
Mensaje: --Connection: Keep-Alive
Mensaje: --
```

Si dejamos tal cual este programa, nos puede ser muy útil en ciertas circunstancias... por ejemplo, como hemos visto en el caso anterior, para comprobar que manda nuestro navegador a un sitio Web (para depurar problemas)

Por ejemplo .... quiero saber que información arrastra mi navegador cuando se conecta a un phpNuke (gestor de presentación de contenidos gratuito creado en php).

Ataco a mi máquina donde tengo phpNuke



Luego me conecto a mi misma máquina al puerto de mi programa ...

```
Mensaje: --GET / HTTP/1.0
Mensaje: --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Mensaje: --Accept-Language: es
Mensaje: --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Mensaje: --Host: localhost:90
Mensaje: --Connection: Keep-Alive
Mensaje: --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Mensaje: --
```

Y sorpresa .... compruebo que me genera un cookie ..... para mantener la sensación de sesión .....

(a algunos seguro que se les ha hecho los ojos chirivitas .....por el posible problema de seguridad cuando navegais de un Web a otro .....) seguro a partir de ahora ... cuando esteis logados en un servidor ..... pinchareis al botón "cerrar sesión" antes de navegar por otros Webs .....

Dotar de funcionalidad nuestro servidor Web

No se si ya habeís tenido la percepción de que nuestro programa tiene algunas deficiencias:

- Primero ... que no vemos que nos pide el usuario y no se lo enviamos .....
- Segundo .... nuestro programa solo sirve una petición y se apaga .....
- Tercero .... como resolveríamos la situación (normal) de que más de un usuario nos realizase peticiones simultaneas .....

He aquí donde ya identificamos ciertas necesidades que vamos a tratar de resolver.

Multiproceso

Vamos por cachos .... primero empezamos por los problemas más graves.... vamos a hacer que nuestra aplicación sea multi-hilo (multi-thread) de tal modo que vamos a crear una nueva clase ... que derive de Thread y que sea capaz de atender cada petición en paralelo que recibamos

Modificamos la clase principal:

```
boolean arranca()
{
	depura("Arrancamos nuestro servidor",DEBUG);

	try
	{
		ServerSocket s = new ServerSocket(90);
		depura("Quedamos a la espera de conexion");

		while(true)  // bucle infinito .... ya veremos como hacerlo de otro modo
		{
			Socket entrante = s.accept();
			peticionWeb pCliente = new peticionWeb(entrante);
			pCliente.start();
		}

	}
	catch(Exception e)
	{
		depura("Error en servidor\n" + e.toString());
	}

	return true;
}
```

Ahora la clase auxiliar

```
class peticionWeb extends Thread
{
	final int ERROR = 0;
	final int WARNING = 1;
	final int DEBUG = 2;

	void depura(String mensaje)
	{
		depura(mensaje,DEBUG);
	}

	void depura(String mensaje, int gravedad)
	{
		System.out.println(currentThread().toString() + " - " + mensaje);
	}

	private Socket scliente  = null;		// representa la petición de nuestro cliente
	private PrintWriter out  = null;		// representa el buffer donde escribimos la respuesta

	peticionWeb(Socket ps)
	{
		scliente = ps;
		setPriority(NORM_PRIORITY - 1); // hacemos que la prioridad sea baja
	}

	public void run() // emplementamos el metodo run
	{
		depura("Procesamos conexion");

		try
		{
			BufferedReader in = new BufferedReader

				(new InputStreamReader(scliente.getInputStream()));


			String cadena = "";

			while (cadena != null || cadena !="")
			{
				cadena = in.readLine();
				if (cadena != null )
				{
					depura("--" + cadena);
				}
			}
		}
		catch(Exception e)
		{
			depura("Error en servidor\n" + e.toString());
		}

		depura("Hemos terminado");
	}

}
```

Si analizamos la respuesta de nuestro Web ante varias peticiones:

```
--------------------Configuration: C:\java\jakarta-tomcat-4.1.12\common\lib roberto--------------------
Mensaje: Arrancamos nuestro servidor
Mensaje: Quedamos a la espera de conexion
Thread[Thread-1,4,main] - Procesamos conexion
Thread[Thread-1,4,main] - --GET / HTTP/1.0
Thread[Thread-1,4,main] - --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Thread[Thread-1,4,main] - --Accept-Language: es
Thread[Thread-1,4,main] - --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Thread[Thread-1,4,main] - --Host: localhost:90
Thread[Thread-1,4,main] - --Connection: Keep-Alive
Thread[Thread-1,4,main] - --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Thread[Thread-1,4,main] - --
Thread[Thread-1,4,main] - Error en servidor
java.net.SocketException: Connection reset
Thread[Thread-1,4,main] - Hemos terminado
Thread[Thread-2,4,main] - Procesamos conexion
Thread[Thread-2,4,main] - --GET /?a=1 HTTP/1.0
Thread[Thread-2,4,main] - --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Thread[Thread-2,4,main] - --Accept-Language: es
Thread[Thread-2,4,main] - --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Thread[Thread-2,4,main] - --Host: localhost:90
Thread[Thread-2,4,main] - --Connection: Keep-Alive
Thread[Thread-2,4,main] - --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Thread[Thread-2,4,main] - --
Thread[Thread-2,4,main] - Error en servidor
java.net.SocketException: Connection reset
Thread[Thread-2,4,main] - Hemos terminado
Thread[Thread-3,4,main] - Procesamos conexion
Thread[Thread-3,4,main] - --GET /?a=3 HTTP/1.0
Thread[Thread-3,4,main] - --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Thread[Thread-3,4,main] - --Accept-Language: es
Thread[Thread-3,4,main] - --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Thread[Thread-3,4,main] - --Host: localhost:90
Thread[Thread-3,4,main] - --Connection: Keep-Alive
Thread[Thread-3,4,main] - --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Thread[Thread-3,4,main] - --
```

Así, es posible que no tengamos la percepción de multi-hilo. Vamos a poner un retardo en cada petición para ver como se intercalan en el log

Modificamos el programa:

```
while (cadena != null || cadena !="")
{
            cadena = in.readLine();
            if (cadena != null )
            {
                        sleep(500);
                        depura("--" + cadena);
            }
            }
}
```

Vemos el Log donde comprobamos la concurrencia .....

```
--------------------Configuration: C:\java\jakarta-tomcat-4.1.12\common\lib roberto--------------------
Mensaje: Arrancamos nuestro servidor
Mensaje: Quedamos a la espera de conexion
Thread[Thread-1,4,main] - Procesamos conexion
Thread[Thread-1,4,main] - --GET /?a=1 HTTP/1.0
Thread[Thread-1,4,main] - --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Thread[Thread-1,4,main] - --Accept-Language: es
Thread[Thread-1,4,main] - --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Thread[Thread-2,4,main] - Procesamos conexion
Thread[Thread-1,4,main] - --Host: localhost:90
Thread[Thread-2,4,main] - --GET /?a=2 HTTP/1.0
Thread[Thread-1,4,main] - --Connection: Keep-Alive
Thread[Thread-2,4,main] - --Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Thread[Thread-1,4,main] - --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Thread[Thread-2,4,main] - --Accept-Language: es
Thread[Thread-1,4,main] - --
Thread[Thread-1,4,main] - Error en servidor
java.net.SocketException: Connection reset
Thread[Thread-1,4,main] - Hemos terminado
Thread[Thread-2,4,main] - --User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
Thread[Thread-2,4,main] - --Host: localhost:90
Thread[Thread-2,4,main] - --Connection: Keep-Alive
Thread[Thread-2,4,main] - --Cookie: ASPSESSIONIDCCTBSQBA=GBDFCHBDFKMNDPBBDFPJIMIO
Thread[Thread-2,4,main] - --
```

Retornar una respuesta

Ahora vamos a incluir la funcionalidad para retornar la página que no solicita el usuario.

```
public void run() // emplementamos el metodo run
{
		depura("Procesamos conexion");

		try
		{
			BufferedReader in = new BufferedReader

				(new InputStreamReader(scliente.getInputStream()));
			out = new PrintWriter

				(new OutputStreamWriter(scliente.getOutputStream(),"8859_1"),true) ;


			String cadena = "";	// cadena donde almacenamos las lineas que leemos
			int i=0;	// lo usaremos para que cierto codigo solo se ejecute una vez

			do
			{
				cadena = in.readLine();

				if (cadena != null )
				{
					// sleep(500);
					depura("--" + cadena + "-");
				}


			 if(i == 0) // la primera linea nos dice que fichero hay que descargar
			 {
			      i++;

			      StringTokenizer st = new StringTokenizer(cadena);

			 if ((st.countTokens() >= 2) && st.nextToken().equals("GET"))
			 {
				retornaFichero(st.nextToken()) ;
			 }
			 else
			 {
				out.println("400 Petición Incorrecta") ;
		}
			   }

			}
			while (cadena != null && cadena.length() != 0);

		}
		catch(Exception e)
		{
				depura("No encuentro el fichero " + mifichero.toString());
			out.println("HTTP/1.0 400 ok");
			out.close();
		}

		depura("Hemos terminado");
}
```

Y añadimos el nuevo método

```
void retornaFichero(String sfichero)
{
		depura("Recuperamos el fichero " + sfichero);

		// comprobamos si tiene una barra al principio
		if (sfichero.startsWith("/"))
		{
			sfichero = sfichero.substring(1) ;
		}

   // si acaba en /, le retornamos el index.htm de ese directorio
   // si la cadena esta vacia, no retorna el index.htm principal
   if (sfichero.endsWith("/") || sfichero.equals(""))
   {
		sfichero = sfichero + "index.htm" ;
   }

   try
   {

		// Ahora leemos el fichero y lo retornamos
		File mifichero = new File(sfichero) ;

		if (mifichero.exists())
		{
				out.println("HTTP/1.0 200 ok");
				out.println("Server: Roberto Server/1.0");
				out.println("Date: " + new Date());
```

```
                                out.println("Content-Type: text/html");
                                out.println("Content-Length: " + mifichero.length());
                                out.println("\n");

                                BufferedReader ficheroLocal = new BufferedReader

                                                (new FileReader(mifichero));


                                String linea = "";

                                do
                                {
                                        linea = ficheroLocal.readLine();

                                        if (linea != null )
                                        {
                                                // sleep(500);
                                                out.println(linea);
                                        }
                                }
                                while (linea != null);

                                depura("fin envio fichero");

                                ficheroLocal.close();
                                out.close();

                        } // fin de si el fiechero existe
                        else
                        {
                                depura("No encuentro el fichero " + mifichero.toString());
                        }
                }
                catch(Exception e)
                {
                        depura("Error al retornar fichero");
                }
}
```



Puede descargarse el código fuente aquí

Sobre el Autor ..

Si desea contratar formación, consultoria o desarrollo de piezas a medida puede contactar con

Creatividad Internet

Autentia S.L. Somos expertos en:
J2EE, C++, OOP, UML, Vignette, Creatividad ..
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
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)
Gestión de contenidos y errores comunes	Os explicamos en que consiste la gestión de contenidos y cuales son los errores cometidos por multitud de empresas a la hora de abordar su implantación
Upload de ficheros en Java	Os mostramos como enviar ficheros a un servidor Web y manipularlos en un servlet en el servidor, gracias a APIs de apache

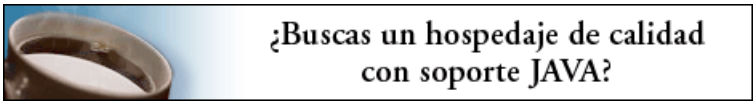
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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



www.AdictosAlTrabajo.com Opimizado 800X600