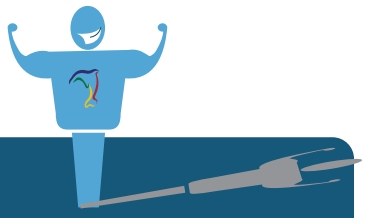
autentia

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...

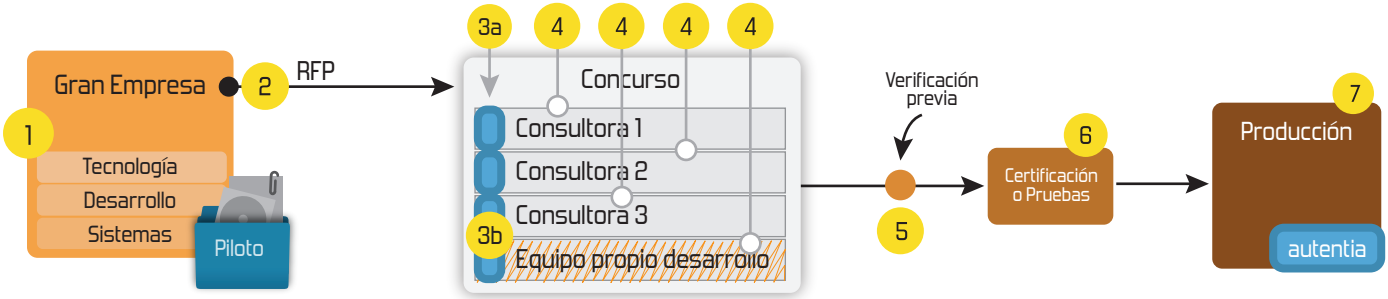
1. Desarrollo de componentes y proyectos a medida



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

adictos altrabajo

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 filtros.pdf

Tomcat Monitoring
Tomcat performance Monitor 5 Tomcats free !

Curso Web J2EE
Curso Avanzado en Desarrollo Web con J2EE

Aquaprojects SL
Tratamientos agua doméstico industr Descalcificacion, Osmosis, etc

Emeco
Depuración de Aguas Residuales Ingeniería Medioambiental

Anuncios Goooooogle Anunciarse en este sitio

Construcción de Filtros en Tomcat

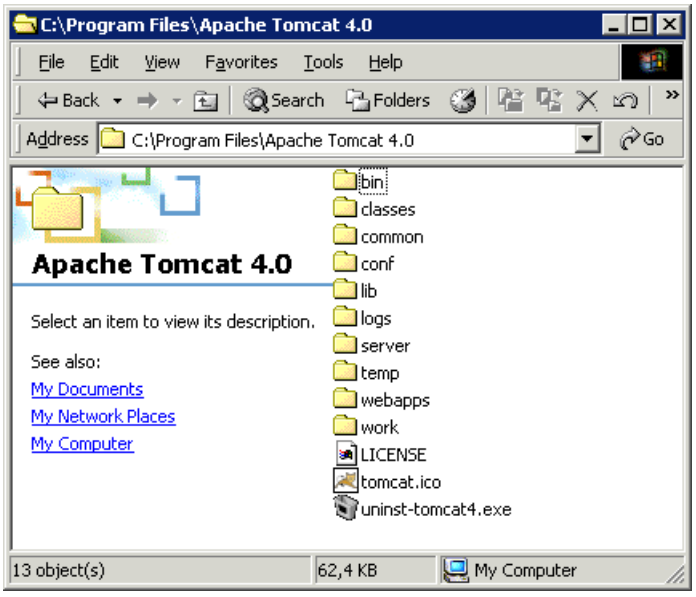
Cuando construimos una aplicación J2EE, muchas veces nos complicamos demasiado la vida resolviendo (normalmente por desconocimiento del entono y falta de tiempo) de un modo complejo problemáticas que ya están de cierto modo resueltas.

Imaginemos que tenemos que realizar una aplicación Web, donde queremos saber el tiempo medio de ejecución de cada una de las distinta peticiones que recibimos (para estadísticas, depuración, etc.). Este mismo ejemplo, posteriormente lo podemos trasladar a otras problematicas, como por ejemplo control de acceso (seguridad) o control de flujo de navegación por nuestro site.

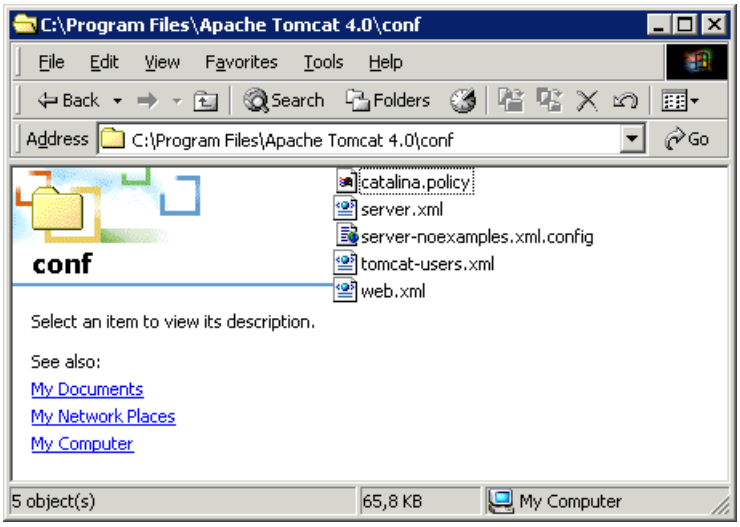
Existen unas piezas en los servidores de aplicaciones que se denominan filtros. Estos filtros, son otro tipo de componente que se puede activar antes de ejecutarse un servlet o path concreto.

Vamos a realizar un pequeño ejemplo pero antes de empezar vamos a crear una nueva WebApp (directorio que contiene todos los recursos de nuestra aplicación) en Tomcat y ver que hay que hacer:

Cuando instalamos Tomcat, vemos que tiene la siguiente estructura.



En el directorio Conf, encontramos los ficheros de configuración de nuestro sistema



En este caso nos interesan 2 de ellos, el **web.xml** que define como se van a comportar todas las aplicaciones que cuelgan de nuestro sistema y el **server.xml** que define estas aplicaciones (y otros parámetros que ya veremos).

En el fichero **server.xml** localizamos esta sección y añadimos nuestro nuevo contexto y decimos (en rojo) que vuelque el log a un fichero.

```
..... aquí hay cosas antes


<!-- Tomcat Root Context -->
<!--
<Context path="" docBase="ROOT" debug="0"/>
-->

<!-- Tomcat Manager Context -->
<Context path="/roberto" docBase="roberto"
debug="0" privileged="true">

<Logger className="org.apache.catalina.logger.FileLogger"
prefix="localhost_roberto_log." suffix=".txt"
timestamp="true"/>
</Context>

<!-- Tomcat Manager Context -->
<Context path="/manager" docBase="manager"
debug="0" privileged="true"/>

..... aquí hay cosas después
```

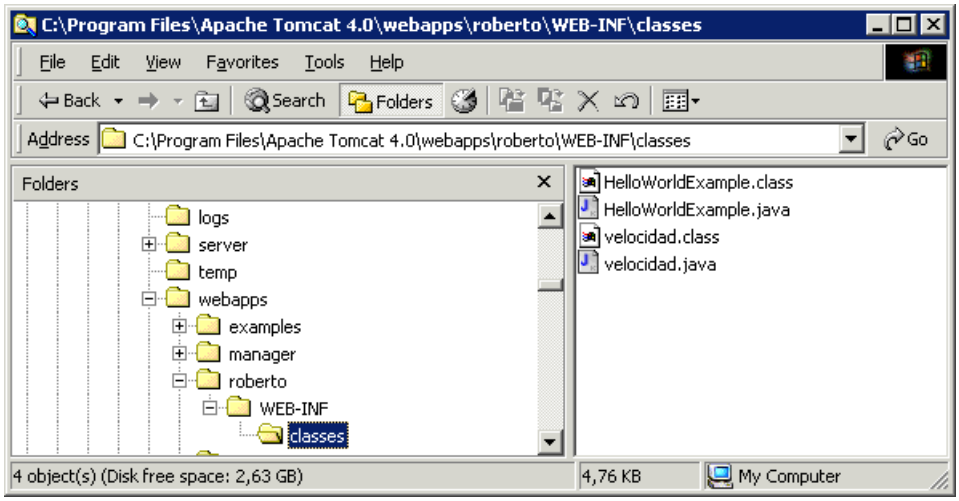
Este nuevo contexto que hemos creado hereda toda la funcionalidad que hay descrita en el **web.xml.** Hemos señalado unas partes significativas de este fichero .....

Con esto comprobamos que todas nuestras WebApps (incluida la que acabamos de crear) tienen soporte ser JSPs y además permite (ver en rojo) cargar servlets dinamicamente si se los metemos colgando del directorio servlet **<url-pattern>/servlet/*</url-pattern>**

```
<servlet>
<servlet-name>invoker</servlet-name>
<servlet-class>org.apache.catalina.servlets.InvokerServlet</servlet-class>
<init-param>
<param-name>debug</param-name>
<param-value>0</param-value>
</init-param>
<load-on-startup>2</load-on-startup>
</servlet>

<!-- The mapping for the invoker servlet -->
<servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>


<!-- The JSP page compiler and execution servlet, which is the mechanism -->
<!-- used by Tomcat to support JSP pages. Traditionally, this servlet -->
<!-- is mapped to URL patterh "*.jsp". This servlet supports the -->
<!-- following initialization parameters (default values are in square -->
<!-- brackets): -->

<servlet>
<servlet-name>jsp</servlet-name>
```

```
<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
<init-param>
<param-name>logVerbosityLevel</param-name>
<param-value>WARNING</param-value>
</init-param>
<load-on-startup>3</load-on-startup>
</servlet>


<welcome-file-list>
<welcome-file>index.html</welcome-file>
<welcome-file>index.htm</welcome-file>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

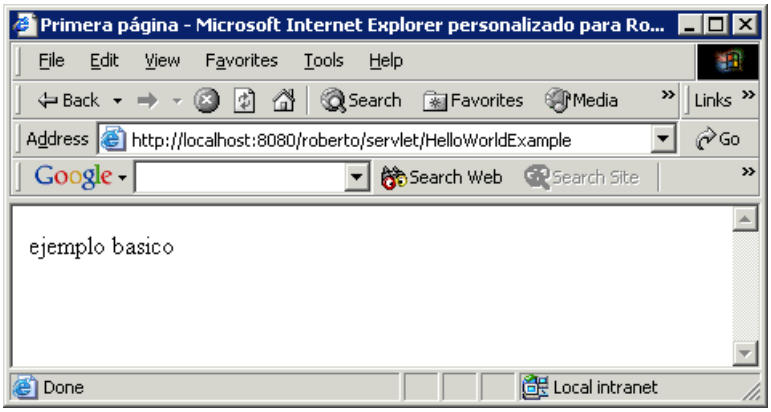
Vemos ahora la estructura de ficheros que hemos creado



Vemos el ejemplo HelloWordExample (reducido del directorio de ejemplos que viene en Tomcat)

```
/* $Id: HelloWorldExample.java,v 1.1.4.1 2001/11/29 18:28:41 remm Exp $
*
*/

import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
* The simplest possible servlet.
*
* @author James Duncan Davidson
*/

public class HelloWorldExample extends HttpServlet {


public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws IOException, ServletException
{
response.setContentType("text/html");
PrintWriter out = response.getWriter();

out.println("<html>");
out.println("<head>");

out.println("<title> Primera página</title>");
out.println("</head>");
out.println("<body bgcolor=\"white\">");
out.println("ejemplo basico");
out.println("</body>");
out.println("</html>");
}
}
```

Lo compilamos e invocamos



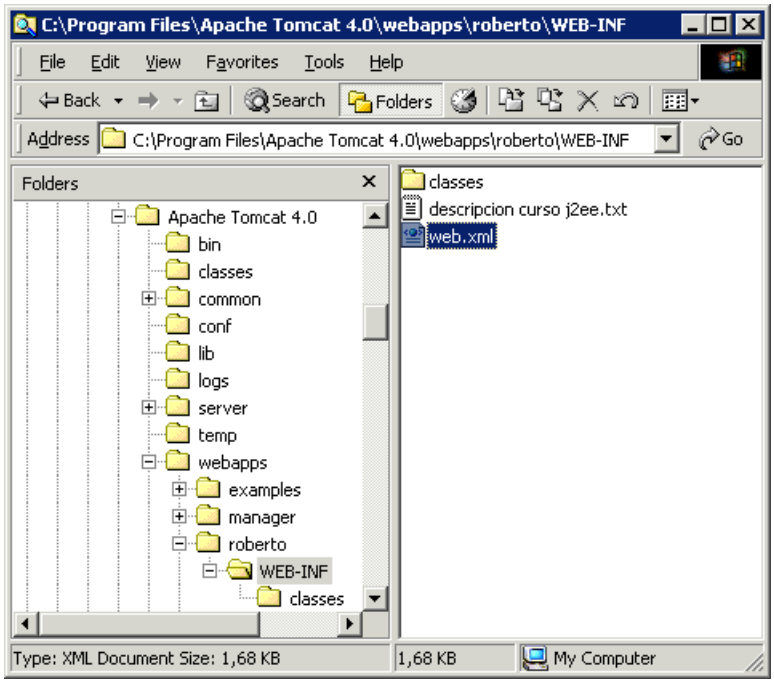
Si analizamos la situación, lo que tenemos es que realmente se ha llamado al servlet **invoker** que nos permite ejecutar los servlets sin necesidad de registrarlos en los ficheros xml (web.xml de nuestra web app)

Ahora vamos a crear un filtro donde vamos a indicar que en nuestra WebApp, antes de llamar al **invoker** se pase un filtro que se llama velocidad (que es una simplificación del **ExampleFilter** que podeis encontrar en los ejemplos de Tomcat)

Creamos el filtro. Lo que hacemos es recoger el tiempo de proceso antes y después de ejecutar la petición de verdad y mostrar en el log, ese dato.

```
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;


public final class velocidad implements Filter
{
            // parametroos de configuracion
            private FilterConfig configuracionFiltro = null;


  public void doFilter(ServletRequest request, ServletResponse response,FilterChain chain) throws IOException, ServletException
            {
                        // Time and log the subsequent processing
                        long comienzo = System.currentTimeMillis();
                        chain.doFilter(request, response);

                        long fin = System.currentTimeMillis();
                        configuracionFiltro.getServletContext().log (this.toString() + ": " + (fin - comienzo) + " milisegundos");
   }


   public void init(FilterConfig pConfiuracionFiltro) throws ServletException
   {
                        configuracionFiltro = pConfiuracionFiltro;
   }

   public String toString()
   {

                        if (configuracionFiltro == null)
                           return ("FiltroTiempo()");

                        StringBuffer sb = new StringBuffer("FiltroTiempo(");
                        sb.append(configuracionFiltro);
                        sb.append(")");
                        return (sb.toString());
   }

   public void destroy()
   {
                configuracionFiltro = null;
   }

}
```

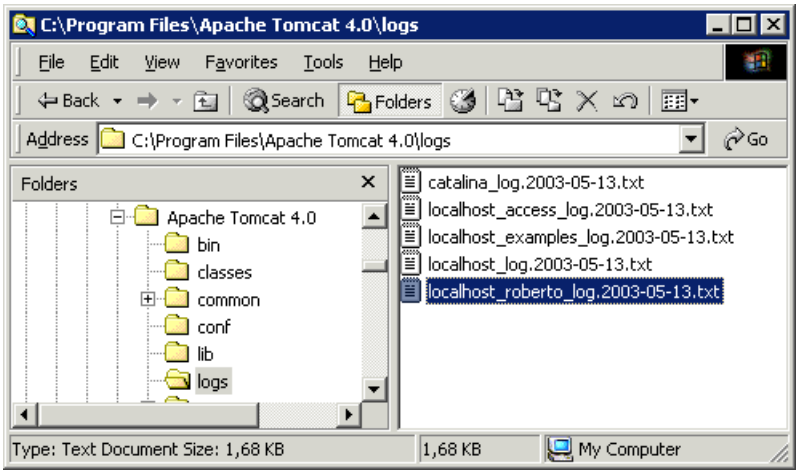
Ahora si tenemos que tocar el web.xml de nuestra WebApp para activar el filtro



En este caso, nuestro fichero es muy sencillo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

<filter>
    <filter-name>Medida del tiempo</filter-name>
    <filter-class>velocidad</filter-class>
</filter>

<filter-mapping>
    <filter-name>Medida del tiempo</filter-name>
     <servlet-name>invoker</servlet-name>
</filter-mapping>

</web-app>
```

Si realizamos ahora de nuevo la petición, veremos en el fichero de logs




```
2003-05-13 14:30:55 WebappLoader[/roberto]: Deploying class repositories to work directory C:\Program
Files\Apache Tomcat 4.0\work\Standalone\localhost\roberto
2003-05-13 14:30:55 StandardManager[/roberto]: Seeding random number generator class
java.security.SecureRandom
2003-05-13 14:30:55 StandardManager[/roberto]: Seeding of random number generator has been completed
2003-05-13 14:30:56 ContextConfig[/roberto]: Added certificates -> request attribute Valve
2003-05-13 14:30:56 ContextConfig[/roberto]: Configured an authenticator for method FORM
2003-05-13 14:30:56 StandardWrapper[/roberto:default]: Loading container servlet default
2003-05-13 14:30:56 StandardWrapper[/roberto:invoker]: Loading container servlet invoker
2003-05-13 14:31:02 FiltroTiempo(ApplicationFilterConfig[name=Medida del tiempo,
filterClass=velocidad]): 20 milisegundos
```

Si habéis percibido la sutileza ..... se abre un nuevo mundo de posibilidades con los filtros en cascada.

Os invito a que miréis el resto de ejemplos que vienen con Tomcat en el directorio Filters .... para ver aspectos más avanzados ... como transformar la entrada y la salida....

Sobre el Autor ..

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

J2EE, EJBs, Struts...

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
JSP´s y Modelo-Vista-Controlador	En este tutorial os enseñamos como crear un JSP, su relación con los servlets y como crear un ejemplo MVC en Tomcat
Gestión errores en JSPs	Os mostramos como realizar ciertas labores intermedias en JSPs: Comentarios, gestión de errores, formateo de fechas y precompilación de ficheros
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollo. Uno de los Framework (entornos) más extendidos es Struts

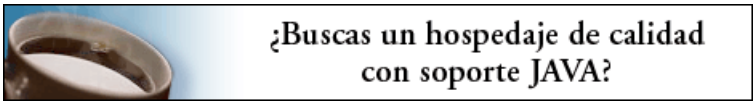
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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