

# ¿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

Tareas programadas (Quartz)  
 Gestor documental (Alfresco)  
 Inversión de control (Spring)

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

BPM (jBPM o Bonita)  
 Generación de informes (JasperReport)  
 ESB (Open ESB)



[Home](#) | [Quienes Somos](#) | [Empleo](#) | [Tutoriales](#) | [Contacte](#)



CoNcept

Lanzado

**TNTConcept versión 0.4.1** ( 04/06/2007)

Desde [Autentia](#) ponemos a vuestra disposición el software que hemos construido (100% gratuito y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (auTeNTia).

Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño ..... Saber más en: <http://tntconcept.sourceforge.net/>

<p><b>Tutorial desarrollado por:</b>  <b>Ángel García Jerez</b> (<a href="#">Autentia</a>) es consultor tecnológico de desarrollo de proyectos informáticos.          Contacta en <a href="mailto:angel.garcia@autentia.com">angel.garcia@autentia.com</a></p>	<p><b>NUEVO CATÁLOGO DE SERVICIOS DE AUTENTIA (PDF 6,2MB)</b></p> <p><a href="http://www.adictosaltrabajo.com">www.adictosaltrabajo.com</a> es el Web de difusión de conocimiento de <a href="http://www.autentia.com">www.autentia.com</a></p>  <p><b>autentia</b> real business solutions</p> <p><a href="#">Catálogo de cursos</a></p>
--	--

Descargar este documento en formato PDF [session\\_listener.pdf](#)

Firma en nuestro libro de Visitas <-----> [Asociarme al grupo AdictosAlTrabajo en eConozco](#)

**Free Eclipse Download**

Eclipse 3.2 and a lot more on demand download configurator  
[www.yoxos.com/ondemand/](http://www.yoxos.com/ondemand/)

**Noaris Innovación**

Desarrollo software Servicios informáticos a medida  
[www.noaris.com](http://www.noaris.com)

**Portal + BPM + ECM**

Gestión unificada de personas, procesos y contenidos  
[www.polymita.com](http://www.polymita.com)

**Cursos de Adobe Premiere**

Conoce esta herramienta al 100% Edición de Vídeo Multimedia y más  
[www.Emagister.com](http://www.Emagister.com)

**Fecha de creación del tutorial: 2007-06-14**

**SESSIONLISTENER: GESTIÓN DE EVENTOS DE SESIÓN EN APLICACIONES WEB.**

1. Introducción
2. [HttpSessionListener](#) y [HttpSessionBindingListener](#) ([javax.servlet.http](#))
3. Demostración
4. Conclusión

**1. Introducción**

Existen determinadas situaciones en proyectos Web en las que necesitamos saber cuando se ha creado/destruido una sesión o cuando se ha añadido/eliminado un objeto de ella. Con este tutorial pretendemos que os familiaricéis con este tipo de operaciones. Para ello vamos a implementar un pequeño ejemplo donde utilizaremos estas técnicas. Nuestra mini-aplicación se encargará de llevar un contador del número de usuarios que actualmente se han conectado a nuestra aplicación y mostrar cuantos de ellos se han logado.

**2. [HttpSessionListener](#) y [HttpSessionBindingListener](#) ([javax.servlet.http](#))**

**HttpSessionListener** y **HttpSessionBindingListener** son dos Interfaces del paquete **javax.servlet.http**. Las clases que las implementen tendrán un comportamiento especial en nuestro contenedor. **HttpSessionListener** contiene dos métodos: **sessionCreated** y **sessionDestroyed** utilizados por el contenedor de aplicaciones para notificarnos cuando una sesión de un usuario ha sido creada o destruida. Aquellas clases que implementen este interfaz deberán ser registradas como listener en nuestro fichero web.xml (en nuestro ejemplo la implementa la clase es.adictos.websession.WebSessionListener). Por otro lado, **HttpSessionBindingListener**, contiene también dos métodos **valueBound** y **valueUnbound** invocados por el contenedor para indicar a la clase que lo implemente que dicho objeto ha sido insertado o eliminado de la sesión (en nuestro ejemplo la implementa la clase es.adictos.websession.bean.User).

Además de estos interfaces, el paquete contiene otros dos **HttpSessionActivationListener** y **HttpSessionAttributeListener**. El primero se utiliza para tener un control cuando utilizamos sesiones distribuidas. Sus métodos serán llamados antes de serializarse (**sessionWillPassivate**) para ser enviados a otra Máquina Virtual, y cuando son deserializados (**sessionDidActivate**) para que la sesión sea utilizada en otra Máquina Virtual. La clase que implemente el segundo deberá ser añadida al fichero web.xml como **HttpSessionListener**. **HttpSessionAttributeListener** contiene tres métodos llamados cuando un objeto es añadido a la sesión (**attributeAdded**), eliminado (**attributeRemoved**) o reemplazado (**attributeReplaced**). No lo hemos comentado pero la gran mayoría de los interfaces comentados están disponibles a partir de la especificación Servlet 2.3.

**3. Demostración**

Los contadores que mantienen el número de usuarios conectados y logados deben estar almacenados en el Contexto del contenedor y cualquiera que intente modificarlos u obtener su valor deberá utilizar una región crítica. Esto es necesario porque pueden producirse situaciones donde varios hilos en paralelo puedan estar operando con ellos lo que puede provocar inconsistencias.

Como hemos comentado antes se ha creado la clase WebSessionListener encargada de mantener el contador del número de usuarios que se encuentran conectados. El método sessionCreated aumentará el contador "usuariosConectados", mientras sessionDestroyed lo disminuirá.

```
package es.adictos.websession;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;
public class WebSessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent arg0) {

        System.out.println("Session creada");
        ServletContext contexto = arg0.getSession().getServletContext();
        synchronized (contexto) {

            Integer usuarioConectados = (Integer)
contexto.getAttribute("usuariosConectados");
            if (usuarioConectados == null) {
                usuarioConectados = new Integer(0);
            }
            usuarioConectados+=1;
            contexto.setAttribute("usuariosConectados",
                usuarioConectados);

        }

    }

    public void sessionDestroyed(HttpSessionEvent arg0) {

        System.out.println("Session destruida");
        ServletContext contexto = arg0.getSession().getServletContext();
        synchronized (contexto) {

            Integer usuarioConectados = (Integer)
contexto.getAttribute("usuariosConectados");
            if (usuarioConectados == null) {
                usuarioConectados = new Integer(0);
            }
            usuarioConectados-=1;
            contexto.setAttribute("usuariosConectados",
                usuarioConectados);

        }

    }

}
```

Por otro lado, tenemos la clase User, un Bean que almacena los datos del usuario y que también se encarga de incrementar (valueBound) o decrementar (valueUnBound) el contador de usuarios logados en la aplicación.

```
package es.adictos.websession.bean;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpSessionBindingEvent;
import javax.servlet.http.HttpSessionBindingListener;
public class User implements HttpSessionBindingListener {

    public void valueBound(HttpSessionBindingEvent arg0) {

        System.out.println("User añadido a la session");
        ServletContext contexto = arg0.getSession().getServletContext();
        synchronized (contexto) {

            Integer usuarioLogados = (Integer)
contexto.getAttribute("usuariosLogados");
            if ( usuarioLogados == null) {
                usuarioLogados = new Integer(0);
            }
            usuarioLogados+=1;
            contexto.setAttribute("usuariosLogados",
                usuarioLogados);

        }

    }

    public void valueUnbound(HttpSessionBindingEvent arg0) {

        System.out.println("User eliminado de session");
        ServletContext contexto = arg0.getSession().getServletContext();
        synchronized (contexto) {

            Integer usuarioLogados = (Integer)
contexto.getAttribute("usuariosLogados");
            if ( usuarioLogados == null) {
                usuarioLogados = new Integer(0);
            }
            usuarioLogados-=1;
            contexto.setAttribute("usuariosLogados",


```

```

        usuarioLogados);
    }
}
}

```

Unido a estas dos clases se han desarrollado tres servlet: Login que almacena el usuario en la sesión, Logout que elimina de la sesión al usuario y Home que muestra el acceso al portal. Todos ellos serán redirigidos a una jsp denominada "estado\_session.jsp" que muestra el estado del número de usuarios conectados/logados y un botón con la acción Login o Logout dependiendo de si el usuario se encuentra en sesión o no. A continuación se muestra la implementación de cada componente:

### Home

```

package es.adictos.websession.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Home extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        doPost(arg0, arg1);
    }
    @Override
    protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        getServletContext().getRequestDispatcher("/estado_session.jsp").forward(arg0,
        arg1);
    }
}

```

### Login

```

package es.adictos.websession.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import es.adictos.websession.bean.User;
public class Login extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        doPost(arg0, arg1);
    }
    @Override
    protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        arg0.getSession().setAttribute("usuario", new User());
        getServletContext().getRequestDispatcher("/estado_session.jsp").forward(arg0,
        arg1);
    }
}

```

### Logout

```

package es.adictos.websession.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Logout extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        doPost(arg0, arg1);
    }
    @Override
    protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1)
    throws ServletException, IOException {
        arg0.getSession().removeAttribute("usuario");
        getServletContext().getRequestDispatcher("/estado_session.jsp").forward(arg0,
        arg1);
    }
}

```

### estado\_session.jsp

```

<% ServletContext contexto = getServletContext();
Integer usuarioConectados= null;
Integer usuarioLogados = null;
synchronized (contexto) {
    usuarioConectados= (Integer) contexto.getAttribute("usuariosConectados");
    usuarioLogados = (Integer) contexto.getAttribute("usuariosLogados");
}
%>
<html>
<head><title>Adictos al trabajo: Estado Session</title></head>
<body>
<div>
<ul>
<li>
Usuario Logados: <%= (usuarioLogados==null) ? "0" : usuarioLogados.toString() %>
</li>
</ul>
</div>

```

```

Usuario Conectados: <%= (usuarioConectados == null) ? "0" : usuarioConectados.toString() %>
</li>
</ul>
</div>
<%
if (session.getAttribute("usuario") == null) {%>
<form action="/WebSession/Login" method="post">
<input type="submit" value="Login">
</form>
<% } else {%>
<form action="/WebSession/Logout" method="post">
<input type="submit" value="Logout">
</form>
<% }%>
</body>
</html>

```

Ahora mostramos el fichero web.xml, en el que se ha reducido al máximo el timeout de la sesión para mostrar la situación en la que el contenedor elimina la sesión por timeout.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

```

```

<display-name>
WebSession</display-name>
<welcome-file-list>

    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>

</welcome-file-list>
<listener>

    <listener-
class>es.adictos.websession.WebSessionListener</listener- class>

</listener>
<servlet>

    <servlet-name>Login</servlet-name>
    <servlet-class>es.adictos.websession.servlet.Login</servlet-
class>< br>

</servlet>
<servlet>

    <servlet-name>Logout</servlet-name>
    <servlet-class>es.adictos.websession.servlet.Logout</servlet-
class>< br>

</servlet>
<servlet>

    <description>
</description>
<display-name>
Home</display-name>
<servlet-name>Home</servlet-name>
<servlet-class>
es.adictos.websession.servlet.Home</servlet-class>

</servlet>
<servlet-mapping>

    <servlet-name>Login</servlet-name>
    <url-pattern>/Login</url-pattern>

</servlet-mapping>

    <servlet-mapping>
<servlet-name>Logout</servlet-name>
<url-pattern>/Logout</url-pattern>

</servlet-mapping>

    <servlet-mapping>

        <servlet-name>Home</servlet-name>
        <url-pattern>/Home</url-pattern>

    </servlet-mapping>

<session-config>
<session-timeout>1</session-timeout>
</session-config>

</web-app>

```

Y por último mostramos un ejemplo de ejecución.



### 3 Conclusiones

Como habéis podido comprobar la captura de eventos relacionados con la sesión es muy sencilla. Lo único que debemos implementar son los interfaces de que disponemos en el paquete `javax.servlet.http`.



This work is licensed under a [Creative Commons Attribution-NonCommercial-No Derivative Works 2.5 License](https://creativecommons.org/licenses/by-nc-nd/2.5/).  
[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](mailto:info@autentia.com)

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....  
**Autentia = Soporte a Desarrollo & Formación**

Gestión de contenidos

[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.

<b>Subscribirse a Novedades</b>	
<b>e-mail</b>	<input style="width: 90%;" type="text"/>
	<input type="button" value="Enviar"/>

---

## Otros Tutoriales Recomendados ([También ver todos](#))

### Nombre Corto

[Generar imagenes desde Servlets](#)

[Control navegación en Servlets](#)

[Uso de JNDI, includes y cookies en Servlets](#)

[Comunicación entre Applets y Servlets](#)

[Filtros de Servlets en Tomcat](#)

[Serialización Servlet-Applet](#)

### Descripción

Os mostramos como generar ficheros GIF desde un servlet java. Util para generar gráficas dinámicas, contadores, etc

Os mostramos como construir el esqueleto de una aplicación basada en Servlets y JSP, con control de navegación.

En este tutorial veremos como usar variables de entorno desde JNDI, incluir un servlet en otro (include) y como usar cookies en Servlets

Os mostramos como comunicar un applet y un servlet a través de GET y POST, serializando objetos y teniendo en cuenta proxys y autenticación

En este tutorial os enseñamos la técnica (poco conocida) del encadenamiento de filtros en la activación de servlets, dentro del entorno Tomcat

Os mostramos un ejemplo para serializar una respuesta en la comunicación servlet-applet

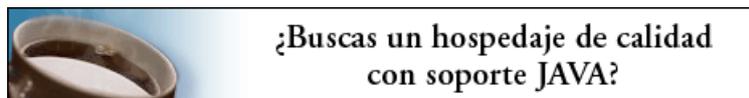
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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](mailto:rcanales@adictosaltrabajo.com) para su resolución.

[Patrocinados por enredados.com .... Hosting en Castellano con soporte Java/J2EE](#)



www.AdictosAlTrabajo.com Optimizado 800X600