

## MÓDULO 7

### EJERCICIOS RESUELTOS LISTENERS

#### TEORIA 1

Explica de forma general qué es un Listener (no es necesario especificar ninguna de forma particular, sólo el concepto de Listener).

Es un servlet que se tiene que definir en el xml descriptor de la aplicación y este servlet está escuchando permanentemente, mientras está activa la aplicación y se ejecuta cuando se produce un determinado evento en la aplicación. Por ejemplo, cuando se inicia una sesión, o cuando se añade un parámetro de sesión, etc.

#### PREGUNTA 1

Crea la plantilla de un listener de nombre MiServletContextListener que implemente la interficie ServletContextListener. Realiza su registro en el descriptor de la aplicación web.xml.

Nota: El listener MiServletContextListener debe registrar en un fichero de texto (que llamaremos RegistroAplicacion.txt) la hora en que se inicia la aplicación, la hora en que termina y el tiempo en minutos que ha estado funcionando. Lo iremos haciendo poco a poco.

```
public class MiServletContextListener implements ServletContextListener{

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

}
```

```
<listener>
    <listener-class>listener.MiServletContextListener</listener-class>
</listener>
</web-app>
```

#### PREGUNTA 2

Crea un parámetro de contexto con el path del fichero RegistroAplicacion.txt y el código necesario que permita leerlo en la función contextInitialized del listener MiServletContextListener.

```
<context-param>
    <param-name>RegistroAplicacion</param-name>
    <param-value>c:\\RegistroAplicacion.txt</param-value>
</context-param>
```

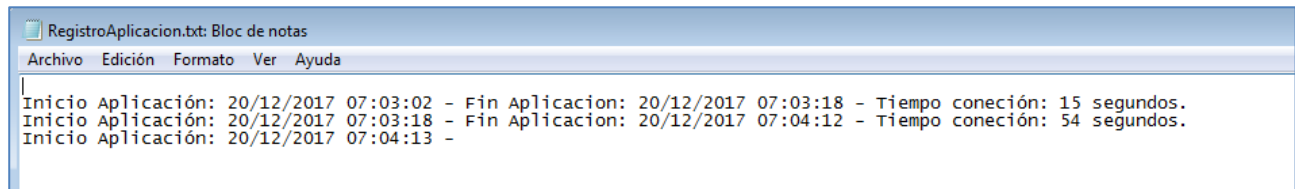
```
public class MiServletContextListener implements ServletContextListener{
    String sFichero;
    long horaInici;

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        ServletContext aplicacion = sce.getServletContext();
        sFichero = aplicacion.getInitParameter("RegistroAplicacion");
    }
}
```

### PREGUNTA 3

Registra en el fichero RegistroAplicacion.txt la hora en la que se inicia la aplicación siguiendo el siguiente formato "Inicio Aplicación: dd/mm/yyyy hh:mm:ss".

NOTA: Declara la variable path del fichero y horaInicial como miembros en la clase, a modo de variable global que sean vistas por los dos métodos.



```
public class MiServletContextListener implements ServletContextListener{
    String sFichero;
    long horaInici;

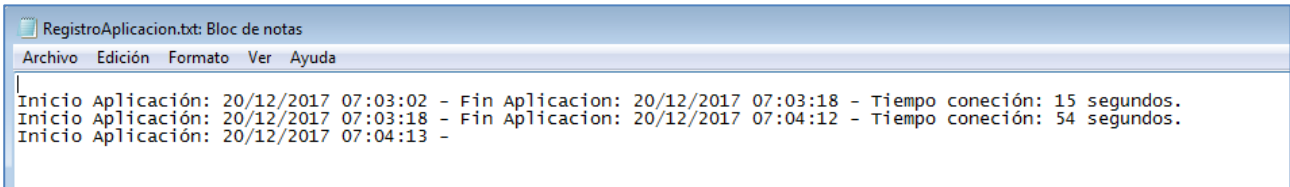
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        ServletContext aplicacion = sce.getServletContext();
        sFichero = aplicacion.getInitParameter("RegistroAplicacion");

        Date now = new Date();
        horaInici = now.getTime();
        String str = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss").format(now);

        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(sFichero, true));
            bw.write("\r\nInicio Aplicación: " + str + " - ");
            bw.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

#### PREGUNTA 4

Registra en el fichero RegistroAplicacion.txt la hora en que termina la aplicación y el tiempo en minutos que ha estado funcionando. Para ello utiliza tanto la variable miembro del path del fichero como la horaInicial para hacer los cálculos.



```
RegistroAplicacion.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Inicio Aplicación: 20/12/2017 07:03:02 - Fin Aplicación: 20/12/2017 07:03:18 - Tiempo conexión: 15 segundos.
Inicio Aplicación: 20/12/2017 07:03:18 - Fin Aplicación: 20/12/2017 07:04:12 - Tiempo conexión: 54 segundos.
Inicio Aplicación: 20/12/2017 07:04:13 -
```

```
@Override
public void contextDestroyed(ServletContextEvent sce) {
    Date now = new Date();
    long horaFi = now.getTime();
    long diferencia = (horaFi - horaInici) / 1000;
    String str = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss").format(now);

    try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(sFichero, true));
        bw.write("Fin Aplicacion: " + str + " - ");
        bw.write("Tiempo conexión: " + diferencia + " segundos.");
        bw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

#### PREGUNTA 5

Investiga cuando la función contextDestroyed es llamada.

- Cuando el WAR está siendo actualizado/borrado
- Cuando el servidor está apagando debido a intervención del administrador
- Cuando el servidor está apagando debido a un error de código.

## PREGUNTA 6

Crea la plantilla de un listener de nombre MiServletContextAttributeListener que implemente la interficie ServletContextAttributeListener. Realiza su registro en el descriptor de la aplicación web.xml.

```
public class MiServletContextAttributeListener implements ServletContextAttributeListener{

    @Override
    public void attributeAdded(ServletContextAttributeEvent scae) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void attributeRemoved(ServletContextAttributeEvent scae) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void attributeReplaced(ServletContextAttributeEvent scae) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

}
```

```
<listener>
    <listener-class>listener.MiServletContextAttributeListener</listener-class>
</listener>
</web-app>
```

## PREGUNTA 7

Visualiza el nombre de los atributos de contexto que se crean al inicio de la aplicación, con un simple System.out.println().

```
public class MiServletContextAttributeListener implements ServletContextAttributeListener{

    @Override
    public void attributeAdded(ServletContextAttributeEvent scae) {
        System.out.println("Atributo creado: " + scae.getName()+" Valor: "+scae.getValue());
    }

    @Override
    public void attributeRemoved(ServletContextAttributeEvent scae) {
        System.out.println("Atributo eliminado: " + scae.getName()+" Valor: "+scae.getValue());
    }

    @Override
    public void attributeReplaced(ServletContextAttributeEvent scae) {
        System.out.println("Atributo modificado: " + scae.getName()+" Valor: "+scae.getValue());
    }

}
```

```

Biblioteca_J2EE (run)  Java DB Database Process  GlassFish Server 4.1.1
Información: Created HTTP listener http-listener-2 on host/port 0.0.0.0:8181
Información: Grizzly Framework 2.3.23 started in: 34ms - bound to [/0.0.0.0:8181]
Advertencia: Instance could not be initialized. Class=interface org.glassfish.grizzly.http.server.AddOn, name=htt
Información: Created HTTP listener http-listener-1 on host/port 0.0.0.0:8080
Información: Grizzly Framework 2.3.23 started in: 11ms - bound to [/0.0.0.0:8080]
Información: Atributo eliminado: com.sun.jsp.tagFileJarUrlsCache Valor: {}
Información: Atributo eliminado: com.sun.appserv.tldlistener.map Valor: {}
Información: Atributo eliminado: org.glassfish.jsp.isStandaloneWebapp Valor: true
Información: Atributo eliminado: org.glassfish.jsp.monitor.probeEmitter Valor: com.sun.enterprise.web.jsp.JspPro
Información: Atributo eliminado: com.sun.appserv.tld.map Valor: {file:/C:/Program%20Files/glassfish-4.1.1/glassf
Información: Atributo eliminado: org.glassfish.servlet.habitat Valor: ServiceLocatorImpl(__HK2_Generated_0,0,233
Información: Atributo eliminado: com.sun.jsp.taglibraryCache Valor: {}
Información: Atributo eliminado: com.sun.appserv.jsp.resource.injector Valor: com.sun.enterprise.web.jsp.ResourceIn
Información: Atributo eliminado: com.sun.faces.AnnotatedClasses Valor: []
Información: Atributo eliminado: com.sun.faces.useMyFaces Valor: false
Información: visiting unvisited references
Información: visiting unvisited references
Información: visiting unvisited references
Información: Atributo creado: org.glassfish.jsp.isStandaloneWebapp Valor: true
Información: Atributo creado: com.sun.appserv.tld.map Valor: {file:/C:/Program%20Files/glassfish-4.1.1/glassfish
Información: Atributo creado: com.sun.appserv.tldlistener.map Valor: {}
Información: Atributo creado: org.glassfish.servlet.habitat Valor: ServiceLocatorImpl(__HK2_Generated_0,0,233548
Información: Atributo creado: com.sun.appserv.jsp.resource.injector Valor: com.sun.enterprise.web.jsp.ResourceIn
Información: Atributo creado: org.glassfish.jsp.monitor.probeEmitter Valor: com.sun.enterprise.web.jsp.JspProbe
Información: Atributo creado: org.apache.catalina.resources Valor: org.apache.naming.resources.ProxyDirContext@f
Información: Atributo creado: com.sun.jsp.tldUriToLocationMap Valor: {}
Información: Atributo creado: com.sun.faces.AnnotatedClasses Valor: []
Información: Atributo creado: com.sun.enterprise.web.WebModule.DeploymentContext Valor: org.glassfish.deploy
Información: Atributo creado: com.sun.enterprise.web.WebModule.isDistributable Valor: false
Información: Atributo creado: com.sun.enterprise.web.WebModule.enableHA Valor: false
Información: Atributo eliminado: com.sun.enterprise.web.WebModule.DeploymentContext Valor: org.glassfish.deploy
Información: Atributo eliminado: com.sun.enterprise.web.WebModule.isDistributable Valor: false
Información: Atributo eliminado: com.sun.enterprise.web.WebModule.enableHA Valor: false
Información: Atributo creado: com.sun.jsp.taglibraryCache Valor: {}
Información: Atributo creado: com.sun.jsp.tagFileJarUrlsCache Valor: {}
Información: Loading application [Biblioteca_J2EE] at [/M7_Biblioteca_J2EE]
Información: Biblioteca_J2EE was successfully deployed in 1.027 milliseconds.

```

## PREGUNTA 8

Después del inicio, en qué punto de la aplicación se vuelve a disparar el evento `attributeAdded` del listener `MiServletContextAttributeListener`.

Una vez se envía el formulario de `index.html` hacia el servlet `GestorBibliotecaServlet`, y después de que éste acaba su ejecución (antes del reenvío hacia `bienvenido.jsp`) se vuelve a disparar el evento `attributeAdded` con el siguiente mensaje:

```
Información: Atributo creado: jsp.x.1st.request Valor: true
```

### PREGUNTA 9

Crea la plantilla de un listener de nombre MiHttpSessionListener que implemente la interficie HttpSessionListener. Realiza su registro en el descriptor de la aplicación web.xml.

```
public class MiHttpSessionListener implements HttpSessionListener{

    @Override
    public void sessionCreated(HttpSessionEvent hse) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent hse) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

```
<listener>
    <listener-class>listener.MiHttpSessionListener</listener-class>
</listener>
</web-app>
```

### PREGUNTA 10

¿En qué puntos se disparará cada uno de sus eventos sessionCreated y sessionDestroyed? Compruébalo.

```
16 public class MiHttpSessionListener implements HttpSessionListener{
17
18     @Override
19     public void sessionCreated(HttpSessionEvent hse) {
20         System.out.println("Inicio de sesión!!");
21     }
22
23     @Override
24     public void sessionDestroyed(HttpSessionEvent hse) {
25         System.out.println("Fin de sesión!!");
26     }
27 }
```

Se ejecuta el evento sessionCreated() cuando en el servlet GestorBibliotecaServlet se ejecuta la siguiente instrucción:

```
HttpSession session = request.getSession();
```

Se ejecuta el evento sessionDestroyed cuando en el servlet ListaUsuarioServlet se ejecuta la instrucción:

```
session.invalidate();
```

También cuando estando en sesión, reiniciamos/modificamos el war (ejecutable J2EE)



## PREGUNTA 11

Crea la plantilla de un listener de nombre `MiHttpSessionAttributeListener` que implemente la interficie `HttpSessionListener`. Realiza su registro en el descriptor de la aplicación `web.xml`.

Nota: Tendremos que hacer un `HttpSessionAttributeListener` que lleve la cuenta de cuántos usuarios hay conectados en el sistema. Deberemos guardar este número en un fichero de texto, que colocaremos en la misma carpeta que el de libros y llamaremos “`usuariosConectados.txt`”

```
public class MiHttpSessionAttributeListener implements HttpSessionAttributeListener {

    @Override
    public void attributeAdded(HttpSessionBindingEvent hsbe) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void attributeRemoved(HttpSessionBindingEvent hsbe) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void attributeReplaced(HttpSessionBindingEvent hsbe) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

}
```

```
<listener>
    <listener-class>listener.MiHttpSessionAttributeListener</listener-class>
</listener>
</web-app>
```

## PREGUNTA 11

Haz que el listener `MiHttpSessionAttributeListener` lleve la cuenta del número de usuarios conectados al sistema. Programa el evento `attributeAdded` para que agregue uno al contador de usuarios conectados en caso de que el atributo de usuario sea “`Usuario`”.

Deberemos guardar este número en un fichero de texto, que colocaremos en la misma carpeta que el resto de ficheros `txt` y llamaremos “`usuariosConectados.txt`”. El formato de cada línea de registro debe de ser parecido al siguiente:

usuariosConectados.txt: Bloc de notes				
Archivo	Edición	Formato	Ver	Ayuda
21/12/2017	05:59:32	Usuarios conectados:	1	
21/12/2017	06:00:58	Usuarios conectados:	2	
21/12/2017	06:01:12	Usuarios conectados:	1	
21/12/2017	06:01:23	Usuarios conectados:	0	

```
public class MiHttpSessionAttributeListener implements HttpSessionAttributeListener {
    int contador = 0;
    String sFichero = "c:\\usuariosConectados.txt";

    @Override
    public void attributeAdded(HttpSessionBindingEvent hsbe) {

        String nombreAtt = hsbe.getName();
        String valorAtt = hsbe.getValue().toString();

        if (nombreAtt.equals("Usuario")) {
            contador++;
            try {
                Date now = new Date();
                String str = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss").format(now);
                BufferedWriter bw = new BufferedWriter(new FileWriter(sFichero, true));
                bw.write(str + " Usuarios conectados: " + contador + "\r\n");
                bw.close();
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

## PREGUNTA 12

Programa el evento attributeRemoved para que descuenta uno al contador de usuarios conectados en caso de que el atributo de usuario sea "Usuario".

```
@Override
public void attributeRemoved(HttpSessionBindingEvent hsbe) {

    String nombreAtt = hsbe.getName();
    String valorAtt = hsbe.getValue().toString();

    if (nombreAtt.equals("Usuario")) {
        contador--;
        if (contador >= 0) {
            try {
                Date now = new Date();
                String str = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss").format(now);
                BufferedWriter bw = new BufferedWriter(new FileWriter(sFichero, true));
                bw.write(str + " Usuarios conectados: " + contador + "\r\n");
                bw.close();
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```