**MÓDULO 7**

**EJERCICIOS RESUELTOS LISTENERS**

**TEORIA 1**

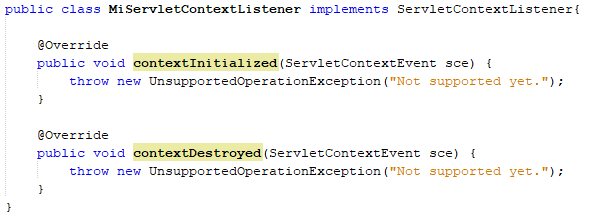
Explica de forma general qué es un Listener (no es necesario especificar ninguna de forma particular, sólo el concepto de Listener).

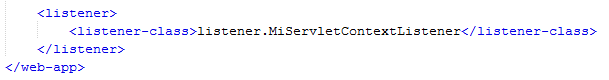
Es un servlet que se tiene que definir en el xml descriptor de la aplicación y este servlet está escuchando permanentemente, mientras está activa la aplicación y se ejecuta cuando se produce un determinado evento en la aplicación. Por ejemplo, cuando se inicia una sesión, o cuando se añade un parámetro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

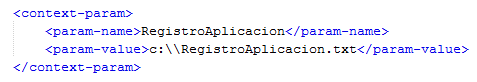
Nota: El listener MiServletContextListener debe registrar en un fichero de texto (que llamaremos RegistroAplicacion.txt) la hora en que se inicia la aplicación, la hora en que termina y el tiempo en minutos que ha estado funcionando. Lo iremos haciendo poco a poco.

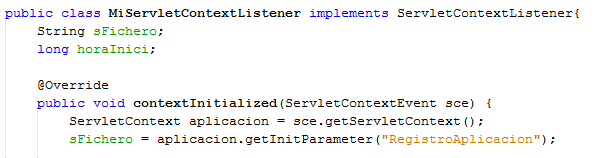




**PREGUNTA 2**

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

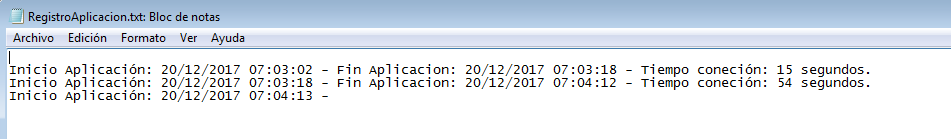


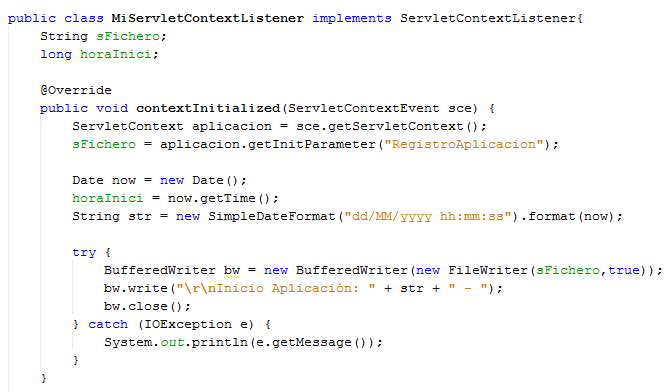


**PREGUNTA 3**

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“.

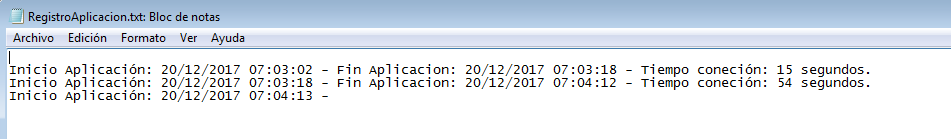
NOTA: Declara la variable path del fichero y horaInicial como miembros en la clase, a modo de variable global que sean vistas por los dos métodos.

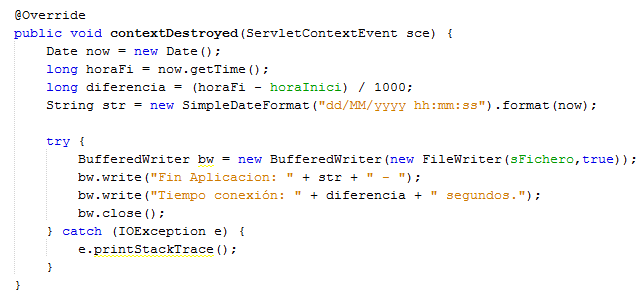




**PREGUNTA 4**

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.





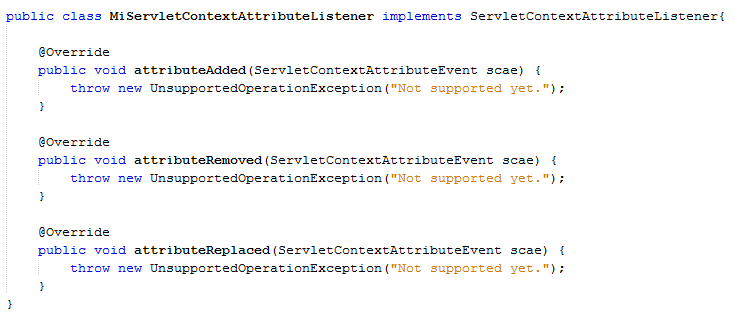
**PREGUNTA 5**

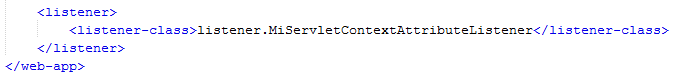
Investiga cuando la función contextDestroyed es llamada.

* 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**

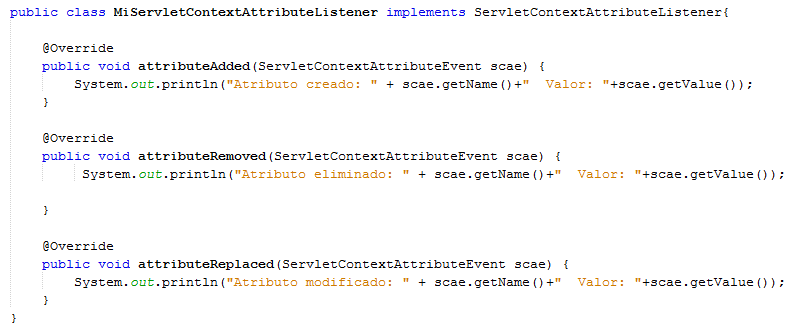
Crea la plantilla de un listener de nombre MiServletContextAttributeListener que implemente la interficie ServletContextAttributeListener. Realiza su registro en el descriptor de la aplicación web.xml.

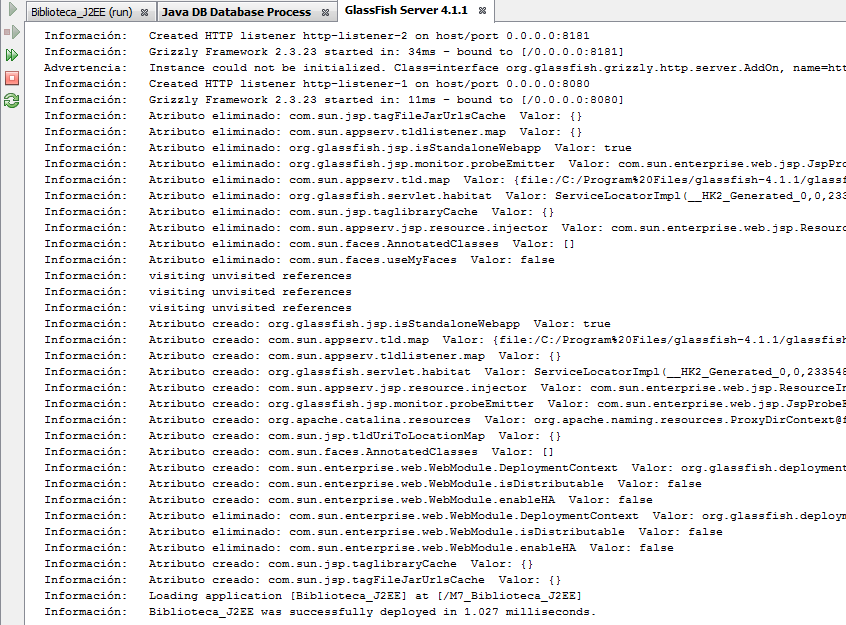




**PREGUNTA 7**

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





**PREGUNTA 8**

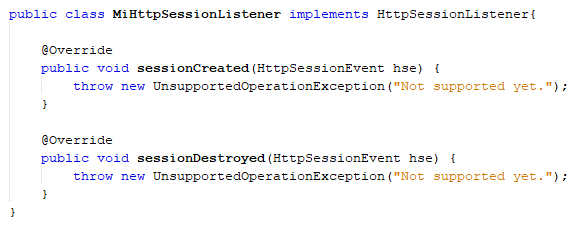
Después del inicio, en qué punto de la aplicación se vuelve a disparar el evento attributeAdded del listener MiServletContextAttributeListener.

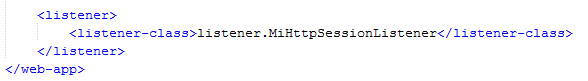
Una vez se envía el formulario de index.html hacia el servlet GestorBibliotecaServlet, y después de que éste acaba su ejecución (antes del reenvío hacia bienvenido.jsp) se vuelve a disparar el evento attributeAdded con el siguiente mensaje:



**PREGUNTA 9**

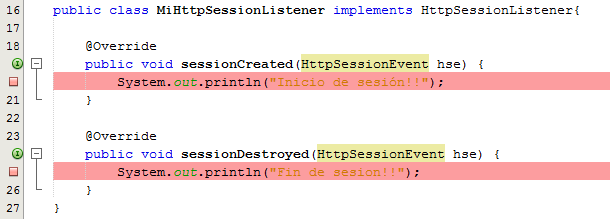
Crea la plantilla de un listener de nombre MiHttpSessionListener que implemente la interficie HttpSessionListener. Realiza su registro en el descriptor de la aplicación web.xml.





**PREGUNTA 10**

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



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 ListaUusarioServlet se ejecuta la instrucción:

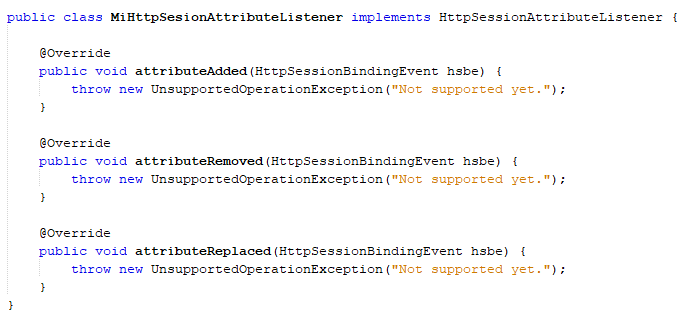
session.invalidate();

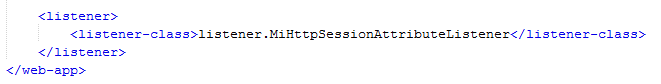
También cuando estando en sesión, reiniciamos/modificamos el war (ejecutable J2EE)

**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”

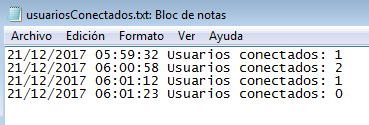


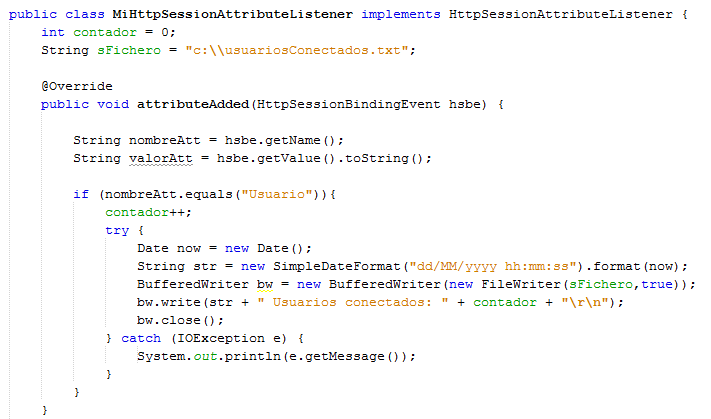


**PREGUNTA 11**

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:





**PREGUNTA 12**

Programa el evento attributeRemoved para que descuente uno al contador de usuarios conectados en caso de que el atributo de usuario sea “Usuario”.

