

► Referencia

1.	<u>INTRODUCCIÓN</u>	3
2.	<u>ESTADO DEL ARTE</u>	4
3.	<u>LOGLOPD</u>	6
3.1	Intercepción a nivel datasource	8
3.2	Intercepción a nivel connectionPoolDatasource	10
3.3	Personalización	11
4.	<u>INSTALACIÓN Y CONFIGURACIÓN</u>	12
4.1	Instalación	12
4.1.1	Librerías	12
4.1.2	Modificar el descriptor de despliegue	12
4.1.3	Spring	13
4.2	Configuración	14
4.2.1	logLOPD.xml	14
5.	<u>EJEMPLO DE LOG GENERADO</u>	17
6.	<u>APLICACIÓN DE EJEMPLO</u>	18
6.1	Configuración	19
6.1.1	Logger	19
6.1.2	Configuración de registro	20
7.	<u>CONCLUSIONES</u>	21
8.	<u>REFERENCIAS</u>	23

1. INTRODUCCIÓN

La Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal, en adelante LOPD, es la Ley que regula el tratamiento de datos personales en todas las empresas, organismos públicos y actividades profesionales.

Desde su entrada en vigor las exigencias legales a las que se tiene que enfrentar toda organización han aumentado de forma considerable, y sumando estas exigencias a la velocidad a la que evolucionan las nuevas tecnologías se hace muy complicado cumplir con todos los requerimientos legales.

Una de las necesidades derivadas del cumplimiento de la LOPD es la de registrar el acceso a aquellos datos de carácter crítico o de nivel alto de seguridad, por lo que bien la aplicación o bien la base de datos debe disponer de un mecanismo que sea capaz de identificar estos accesos y registrarlos. Sin embargo, y como se verá a lo largo de este documento, en ambos casos se presentan diversos problemas de difícil resolución.

Para resolver todos los problemas que plantea la LOPD se presenta *LogLOPD*, que siendo transparente a los desarrolladores y con una instalación sencilla, resuelve de manera efectiva todos los problemas derivados del cumplimiento de la Ley en las aplicaciones basadas en JDBC.

2. ESTADO DEL ARTE

Los registros de accesos para la LOPD tienen que aportar toda la información relacionada con la consulta:

- Consulta SQL completa (identifica tipo de acceso, tabla accedida...)
- Resultados obtenidos por la consulta
- Identidad del usuario que ha ejecutado la consulta
- Fecha y hora en la que se realiza
- Autorizada/Denegada

Las dificultades con las que nos encontramos al intentar cumplir con estos requisitos de audición en base de datos (monitorizando las tablas) son las siguientes:

- Los registros generados por la base de datos no incluyen los valores de los parámetros de las consultas en muchos casos, imposibilitando saber cuales son los datos a los que accede.
- La identidad que aportan los registros de la base de datos es la del usuario de base de datos y no la del usuario que esta accediendo a los datos (usuario de aplicación o identidad del usuario).

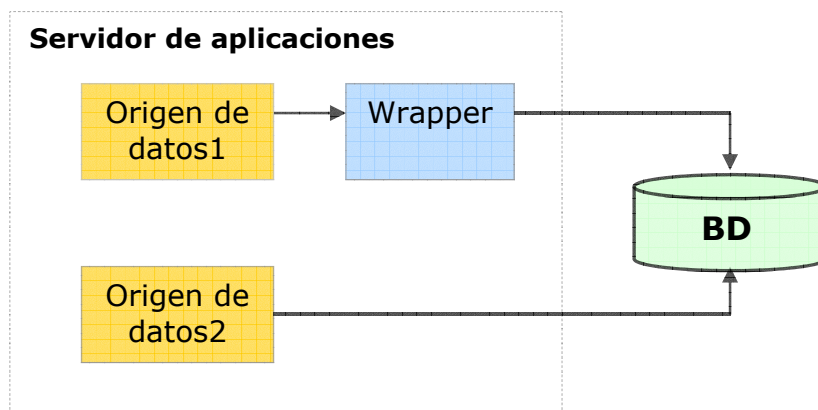
Por otro lado, las dificultades con las que nos encontramos al intentar cumplir con estos requisitos de audición a nivel de aplicación son las siguientes:

- Se necesitan clases que controlen los accesos a base de datos y registren las consultas, lo que implica crear una solución propietaria que no respeta el uso del API JDBC, y que interfiere en una buena arquitectura de la aplicación.
- En el caso de programar manualmente las consultas de registro de acceso en el propio código de la aplicación, se implica a los desarrolladores en la LOPD, no siendo transparente, y siendo difícil detectar errores de omisión de consultas.
- En ningún caso es una solución adaptable a proyectos existentes.

Es en este punto donde entran en juego las funcionalidades ofrecidas por *LogLOPD* para aplicaciones basadas en JDBC:

- Permite registrar todas las consultas SQL realizadas contra la base de datos, incluyendo los valores de los parámetros.

- En los registros se incluye el usuario real que está accediendo a los datos (usuario de aplicación).
- No es necesario modificar el código fuente de las aplicaciones siendo aplicable en aquellas aplicaciones previamente desarrolladas.
- Se aplica a nivel origen de datos (usuario de base de datos) siendo posible auditar únicamente la actividad de un usuario de base de datos y no de toda la base de datos mejorando el rendimiento de las aplicaciones.



El *origen de datos2* no es auditado mientras que el *origen de datos1* sí.

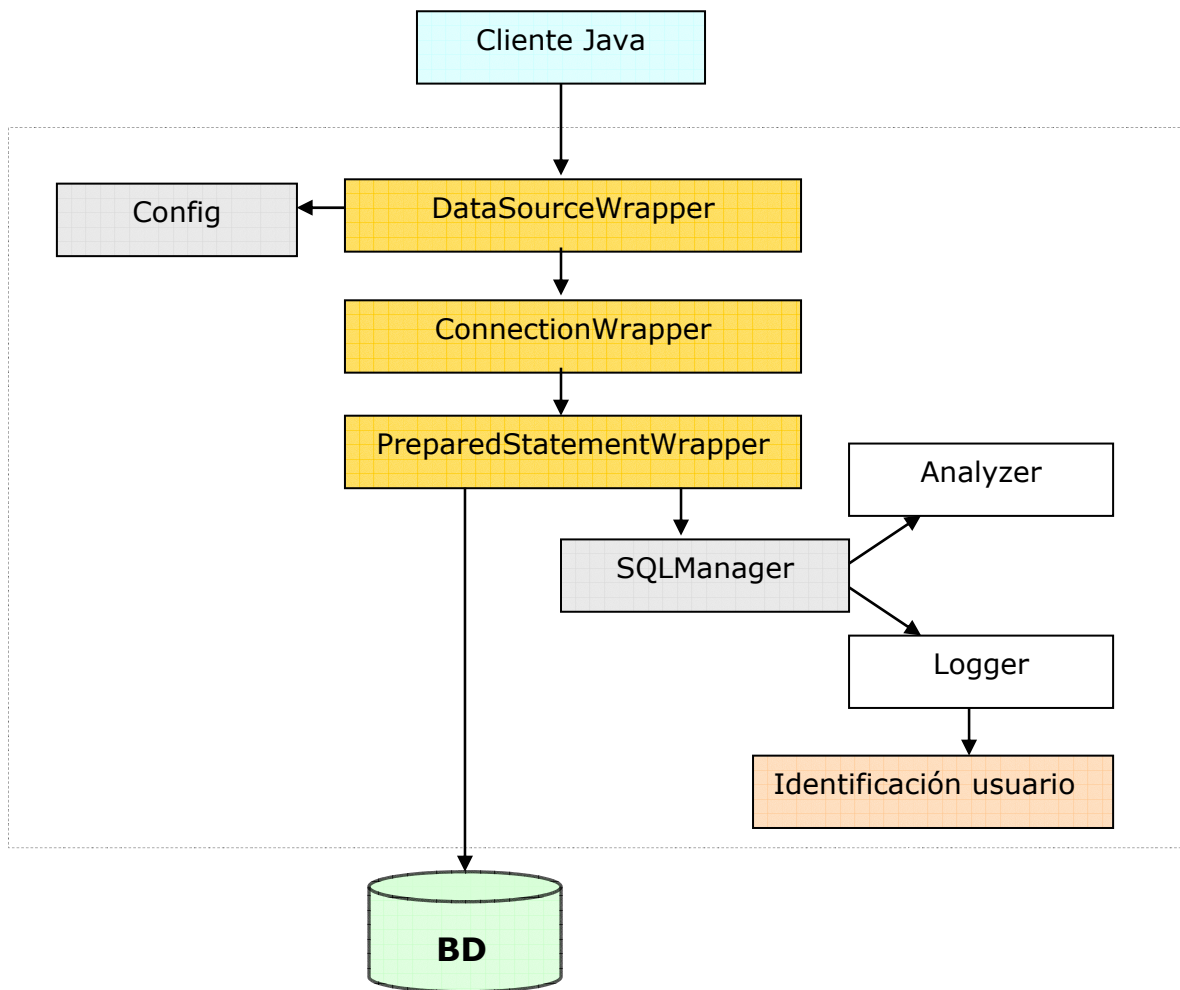
- Es posible, vía configuración, definir las tablas y columnas que se quieren auditar, y si se quieren registrar los resultados devueltos por las consultas. Se puede especificar una configuración para cada origen de datos.

3. LOGLOPD

LogLOPD es un Wrapper del DataSource (origen de datos) utilizado por la aplicación. Dicho de otro modo, envuelve el DataSource de la aplicación manteniendo su comportamiento original y extendiéndolo con nuevas funcionalidades.

Los objetivos de *LogLOPD* son tres:

- Interceptar la consulta SQL
- Analizar la consulta SQL
- Generar un registro en función de los datos accedidos en la consulta y las políticas establecidas por la LOPD, incluyendo la identidad del usuario en el registro (DNI o nombre de usuario por ejemplo).



La idea se basa en utilizar el `DataSourceWrapper`, que encapsulando el `DataSource` real, intercepta las consultas y las analiza para generar un registro en los casos necesarios.

Los módulos de los que se compone *LogLOPD* son los siguientes:

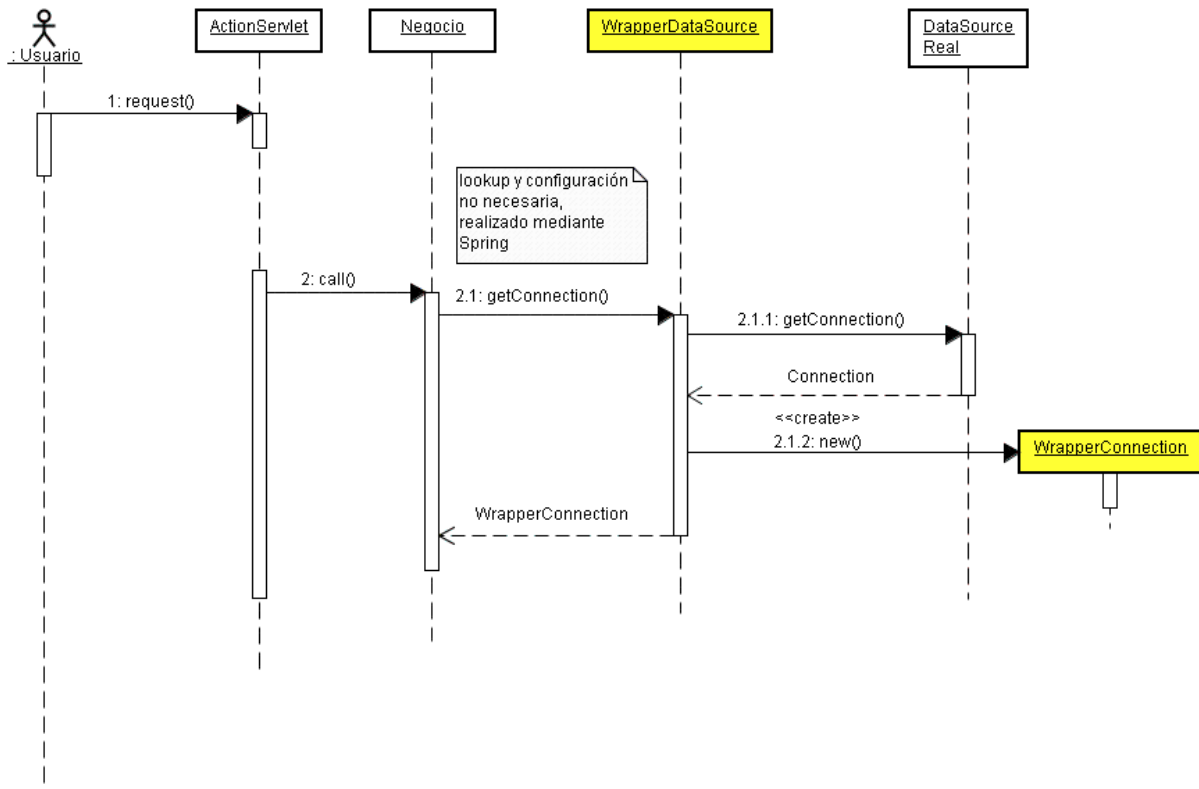
- 1. CLASES WRAPPER:** *DataSourceWrapper*, *PreparedStatementWrapper*, etc... son las clases que implementan el API JDBC, añadiéndoles una funcionalidad extra para el registro y análisis de las consultas.
- 2. CONFIG:** Es el módulo que contiene la configuración de tablas y columnas que contienen datos de carácter personal.
- 3. SQLMANAGER:** Es el encargado de gestionar las llamadas al analyzer y logger. Su funcionalidad radica en ofrecer métodos de tratamiento de las consultas, que las analicen y llamen al logger para registrar la consulta con sus datos asociados.
- 4. ANALYZER:** Es el encargado de analizar consultas, conforme a la configuración del `DataSource`, y devolver el resultado del análisis.
- 5. LOGGER:** Es el módulo que se encarga de guardar los registros.
- 6. IDENTIFICACIÓN USUARIO:** El logger recibe el usuario real de la aplicación, de forma que en el registro de consultas quedará registrado dicho usuario.

En *LogLOPD* se establecen dos posibles opciones de intercepción del `DataSource` real.

3.1 Intercepción a nivel datasource

En este caso la aplicación consume el DataSourceWrapper en vez del DataSource real. Las conexiones creadas por el DataSourceWrapper son del tipo ConnectionWrapper, por lo que aportarán la funcionalidad derivada del uso de LogLOPD.

En la siguiente imagen podemos ver el flujo de acciones provocado por una clase que accede a una base de datos:



La principal ventaja de este nivel de intercepción es que su configuración utilizando el framework *Spring* es muy sencilla, ya que es posible realizarla de manera declarativa. Es la manera más aconsejable de utilizar y configurar LogLOPD.

Ejemplo:

```
<bean id="dataSourceWrapper"
      class="com.eurohelp.wrapperjdbc.wrapperlopd.WrapperLOPDDataSource">
  <property name="ds" ref="dataSource"/>
  <property name="sqlmanager" ref="sqlmanager"/>
  <property name="config" ref="config"/>
</bean>

<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/dataSourceRealDS"/>
```

En los casos que no se utilice *Spring* la intercepción se debe realizar en la clase que gestione las conexiones con la base de datos (generalmente es una factoría de conexiones *-ConnectionFactory-*), envolviendo el objeto *DataSource* real en un *DataSourceWrapper*.

Ejemplo:

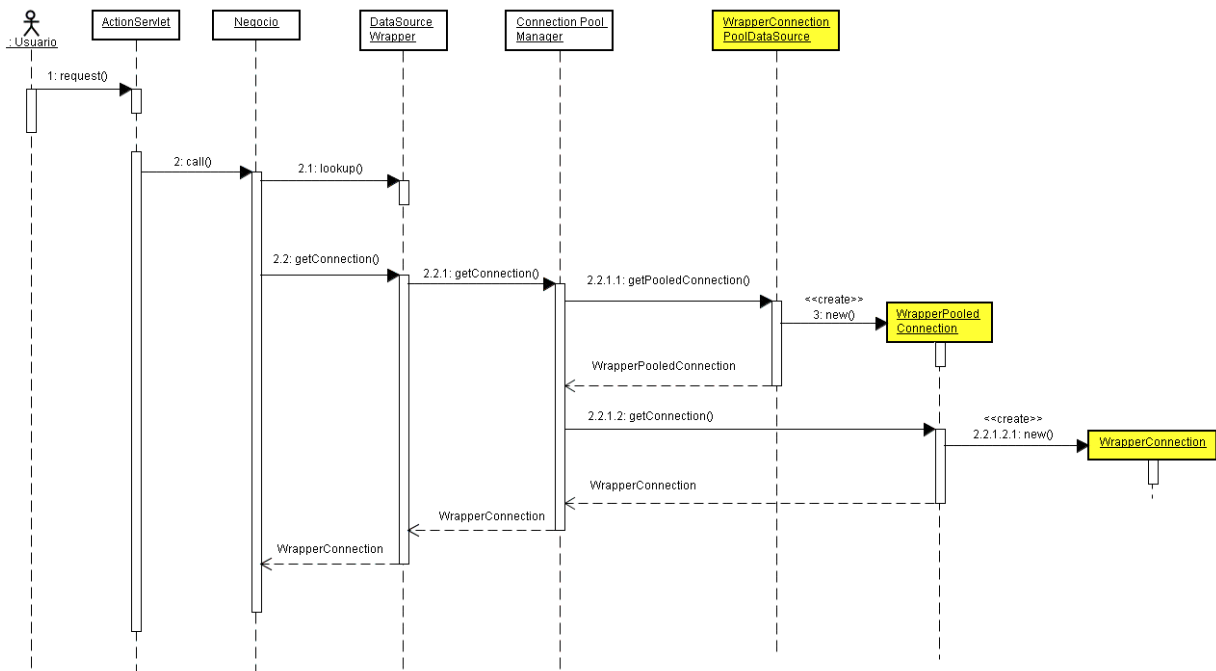
```
Context ctx = new InitialContext();
IConfig config = new WrapperConfig();
config.setTablas(...);
...
ISQLManager manager = new WrapperSQLManager();
DataSource ds = ctx.lookup("java:comp/env/DataSourceRealDS");
DataSource dsWrap = new WrapperLOPDDataSource(ds,manager,config);
Connection cn = dsWrap.getConnection();
...
```

3.2 Intercepción a nivel connectionPoolDataSource

En este caso la intercepción se realiza al configurar el DataSource en el servidor, a nivel ConnectionPoolDataSource.

Todos los objetos del API JDBC consumidos por el cliente implementan los interfaces estándar de dicho API, pero en realidad se trata de Wrappers de las implementaciones originales.

En la siguiente imagen podemos ver el flujo de acciones provocado por esta intercepción.



Los pasos a realizar para configurarlo son:

- Configurar el origen de datos o DataSource real de la aplicación.
- Configurar el DataSource del Wrapper, para que lo utilice la aplicación.

3.3 Personalización

LogLOPD es un producto personalizable gracias al uso de la factoría proporcionada por *Spring*. Esta característica nos permite aplicar una de las bases de la programación orientada a objetos, sustituir una implementación por otra, posibilitando adaptarse a cualquier necesidad propietaria. Especialmente interesante es la implementación del interface *Logger* puesto que permite adaptarse a las necesidades particulares de registro que puedan surgir (registro en base de datos, registro en fichero, en consola...).

4. INSTALACIÓN Y CONFIGURACIÓN

Estos son los pasos para configurar e instalar el Wrapper en una aplicación Web.

4.1 Instalación

La instalación se realiza en 3 pasos:

4.1.1 Librerías

Incluir las siguientes librerías en el classpath de la aplicación (WEB-INF/lib o en el classpath del servidor):

- Wrapper: *wrapper.jar*
- Spring: *spring.jar*

4.1.2 Modificar el descriptor de despliegue

Esta configuración debe ser añadida en el descriptor de despliegue:

- Fichero de configuración de Spring:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/logLOPD.xml
  </param-value>
</context-param>
```

- Inicializar el listener de Spring:

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

- Incluir el filtro de LogLOPD:

```
<filter>
  <filter-name>Filter</filter-name>
  <display-name>Filter</display-name>
  <description>Filtro para LogLOPD</description>
  <filter-class>
    com.eurohelp.wrapperlopd.filter.Filter
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>Filter</filter-name>
  <url-pattern>/Filter</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>Filter</filter-name>
  <servlet-name>action</servlet-name>
</filter-mapping>
```

El filtro debe filtrar todas las peticiones a la aplicación Web. En el caso de ejemplo, se muestra la captura a los actions de una aplicación Struts.

4.1.3 Spring

Añadir *logLOPD.xml* en el directorio WEB-INF de la aplicación Web.

4.2 Configuración

La configuración se define en el fichero *logLOPD.xml* que será consumido por Spring.

4.2.1 logLOPD.xml

- *Logger*

```
<bean id="logger" class="com.eurohelp.wrapperjdbc.logger.DefaultLogger"
>
  <property name="fichero">
    <value>c:\log.txt</value>
  </property>
</bean>
```

En este apartado se ha configurado el logger. La clase debe implementar la interfaz *ILogger* incluida en el *LogLOPD*.

- *SQL Manager*

```
<bean id="sqlmanager" class="
com.eurohelp.wrapperjdbc.sqlmanager.WrapperSQLManager">
  <property name="dataBaseType">
    <value>ORACLE</value>
  </property>
  <property name="logger">
    <ref local="logger"/>
  </property>
</bean>
```

En este apartado se ha configurado el gestor (SQLManager), al que se le indica que logger debe utilizar para registrar las consultas, y la base de datos que utiliza la aplicación. Los valores posibles son ORACLE, DB2, y SQL en caso de otra.

- *Configuración de tablas/columnas y resultados*

```
<bean id="config" class=" com.eurohelp.wrapperjdbc.config WrapperConfig">
  <property name="resultados">
    <value>true</value>
  </property>
  <property name="tablas">
    <map>
      <entry>
        <key>
          <value>TABLA1</value>
        </key>
        <list>
          <value>COLUMNA1</value>
          <value>COLUMNA2</value>
        </list>
      </entry>
      <entry>
        <key>
          <value>TABLA2</value>
        </key>
        <null/>
      </entry>
    </map>
  </property>
</bean>
```

En la figura anterior, se está indicando que la COLUMNA1 y COLUMNA2 de la TABLA1, así como la TABLA2, contienen datos de carácter personal. Por lo que se registrarán los accesos/modificaciones a dichas tablas y columnas. Además, se indica que se registren los resultados devueltos por las consultas.

- *Wrapper del DataSource*

```
<bean id="dataSourceWrapper"
class="com.eurohelp.wrapperjdbc.wrapperlopd.WrapperLOPDDataSource">
  <property name="ds" ref="dataSource"/>
  <property name="sqlmanager" ref="sqlmanager"/>
  <property name="config" ref="config"/>
</bean>
<jee:jndi-lookup id="dataSource"
  jndi-name="java:comp/env/dataSourceReal"/>
```

La aplicación debe consumir el bean *dataSourceWrapper* de *Spring* para obtener conexiones con la base de datos. Se debe definir cuál es el DataSource real para inyectarlo en *dataSourceWrapper*.

- *Usuario de la aplicación*

```
<bean id="application_user" class="jpetstore.user.UserImpl"/>
```

El bean *application_user* debe pertenecer a una clase que implemente la interfaz *IUser*, para poder obtener el usuario de la aplicación.

5. EJEMPLO DE LOG GENERADO

Como resultado de la configuración anterior, incluyendo tablas y columnas de base de datos reales, los resultados obtenidos en una aplicación utilizando el logger por defecto y escritura en "c:\log.txt", son los siguientes:

```
31/01/07 10:16:33 USUARIO_APLICACION
SELECT CATID, NAME, DESCN FROM CATEGORY WHERE CATID = 'FISH'

31/01/07 10:19:45 USUARIO_APLICACION
SELECT PRODUCTID, NAME, DESCN, CATEGORY FROM PRODUCT WHERE PRODUCTID = 'K9-PO-
02'
```

log.txt

6. APLICACIÓN DE EJEMPLO

Junto a la distribución de *LogLOPD* se incluye una aplicación de ejemplo; *jpetstore*. Esta aplicación puede ser configurada de diferentes maneras para comprobar el funcionamiento de *LogLOPD*.

Estos son los pasos para instalar la aplicación:

1. En función del servidor de aplicaciones, obtener el archivo .EAR o .WAR
2. Desplegar el archivo en el servidor de aplicaciones

En algunos servidores de aplicaciones J2EE como Tomcat, se debe copiar el archivo .WAR en el directorio *webapps* de la carpeta de instalación. En otros casos, como Rational ApplicationDeveloper 6.0, se debe añadir el archivo .EAR como una nueva aplicación utilizando la consola administrativa.

3. Arrancar la base de datos HyperSonic (*server.bat*)
4. Iniciar el servidor de aplicaciones
5. Ejecutar la página de inicio de la aplicación Web. Se debe conocer el dominio y el puerto donde está corriendo el servidor de aplicaciones.

<http://<dominio:puerto>/jpetstore>

6.1 Configuración

jpetstore está configurado para registrar únicamente las consultas en un el fichero "C:/log.txt", y utilizar un analizador de SQL estándar, ya que utiliza una base de datos HyperSonic.

El registro está definido a nivel de tabla, es decir, se han definido todas las tablas a nivel de configuración como de carácter personal, y ninguna columna dentro de dichas tablas, por lo que se registrarán todos los accesos a todas las tablas.

Se puede modificar la configuración por defecto de la siguiente forma en el fichero *logLOPD.xml* de la carpeta *WEB-INF*:

6.1.1 Logger

El logger puede ser configurado para registrar en un fichero diferente:

```
<bean id="logger" class="com.eurohelp.wrapperjdbc.logger.DefaultLogger" >
  <property name="fichero">
    <value>PATH_AL_NUEVO_FICHERO</value>
  </property>
</bean>
```

O también puede configurarse para mostrar resultados por pantalla:

```
<bean id="logger" class="com.eurohelp.wrapperjdbc.logger.DefaultLogger">
</bean>
```

Así mismo también es posible implementar otra clase logger (que cumpla el interfaz *ILogger*), y utilizarla en la definición.

```
<bean id="logger" class="CLASE_DE_IMPLEMENTACION" >
...
...
</bean>
```

6.1.2 Configuración de registro

Se puede modificar la configuración para que registre, además de las consultas, los resultados leídos de dichas consultas. Así como quitar tablas de la configuración para no registrar los accesos a las mismas.

```
<bean id="config" class="com.eurohelp.wrapperjdbc.config WrapperConfig">
  <property name="resultados">
    <value>true</value>
  </property>
  <property name="tablas">
    <map>
      <entry>
        <key>
          <value>PRODUCT</value>
        </key>
        <null/>
      </entry>
    </map>
  </property>
</bean>
```

7. CONCLUSIONES

El problema derivado del cumplimiento de la LOPD puede ser abordado con dos políticas diferentes.

Una solución propietaria, programando las consultas que registran los accesos en la aplicación, puede ser una solución aceptable. Sin embargo, esto deja en manos de los desarrolladores el cumplimiento de la Ley, pudiendo ser problemático por el hecho de no registrar alguna consulta, y en tiempo de desarrollo, ya que se necesita más tiempo para adaptar la aplicación a esta Ley, impidiendo además adaptar esta solución a proyectos existentes.

La solución alternativa, en base de datos, presenta los problemas que se han explicado con anterioridad, ya que algunas bases de datos no registran los parámetros, y no se puede obtener el usuario de la aplicación que ha realizado la consulta, sino el usuario de base de datos.

LogLOPD solventa los problemas derivados de este cumplimiento, siendo transparente a los desarrolladores, y registrando las consultas con los parámetros y usuario real de la aplicación.

En resumen, las **ventajas ofrecidas por LogLOPD** respecto a las soluciones existentes en la actualidad son las siguientes:

- ✓ Todas las consultas contra la base de datos pasan por el Wrapper sin posibilitar la existencia de accesos no registrados.
- ✓ El código fuente de las aplicaciones queda libre de la responsabilidad de manejar las exigencias de la LOPD, reduciendo los costes de desarrollo y facilitando el trabajo a los equipos de desarrollo.
- ✓ La solución es aplicable a aplicaciones previamente desarrolladas sin modificar el código fuente original.
- ✓ Se adapta a los posibles cambios de la LOPD.
- ✓ Se adapta a cualquier API de acceso a datos basada en JDBC (Hibernate, iBATIS, SpringJDBC, JPA, ...).
- ✓ Es posible definir mediante configuración que datos queremos auditar: tablas, columnas, orígenes de datos...
- ✓ Es posible registrar los resultados devueltos por las consultas.

8. REFERENCIAS

[1]. Spring FrameWork

<http://www.springframework.org>

[2]. API JDBC

<http://java.sun.com/javase/technologies/database/index.jsp>

[3]. LOPD

https://www.agpd.es/upload/Canal_Documentacion/legislacion/Estatad/Ley%2015_99.pdf