Conexión a la base de datos

Se trata de la primera funcionalidad que se debe incorporar a la aplicación: la posibilidad de obtener una conexión con una base de datos concreta. En este trabajo nos centraremos en la obtención directa de conexiones a través de la clase DriverManager, sin utilizar datasources definidos a nivel del servidor web ni pool de conexiones.

Según el API de JDBC, existen tres opciones de obtener conexiones a la base de datos a través del método DriverManager.getConnection()

* getConnection(String url, Properties info)
* getConnection(String url, [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) user, String password)
* getConnection(String url)

En todos los casos es necesario indicar la url que indica siempre el driver que se debe utilizar por parte del servidor para establecer la conexión con la base de datos.

La tercera opción, solo indicar url, no se va a utilizar, puesto que siempre se indicará al menos usuario y contraseña.

La aplicación deberá obtener una conexión a la base de datos cuando el usuario se identifica mediante usuario y contraseña. La conexión se almacenará en la sesión del servidor web y será utilizada en cada petición o consulta que se realice a la base de datos. Cuando la sesión web del usuario muera por inactividad, el proceso correspondiente deberá cerrar la conexión establecida con la base de datos si aún estuviera activa.

Antes de poder utilizar una conexión recuperada de la sesión web del usuario deberá comprobarse que la conexión continúa activa. Si la conexión ya no estuviera activa, se intentará obtener una nueva conexión a la base de datos utilizando para ellos los datos de conexión almacenados en la sesión web del usuario.

# Identificación

La primera pantalla que se presenta al usuario consiste en la pantalla de identificación donde el usuario deberá introducir los datos necesarios para que se establezca la conexión con una base de datos.

Estos datos serán:

* url: será la url de la base de datos, de la forma jdbc.subprotocolo:subnombre
* usuario: nombre del usuario de base de datos
* password: contraseña del usuario de base de datos. La contraseña deberá ir oculta en un campo html tipo password

## web.xml

Carpeta WEB-INF

Toda solicitud realizada para presentar esta página deberá ser redirigida al puerto SSL correspondiente utilizando a partir de entonces el protocolo HTTPS para cualquier comunicación entre cliente y servidor. Para conseguir esta operación, independientemente de la configuración del servidor, debemos definir a nivel de aplicación, la configuración de seguridad de la siguiente manera:

<security-constraint>

<display-name>Constraint1</display-name>

<web-resource-collection>

<web-resource-name>Todos</web-resource-name>

<description/>

<url-pattern>/\*</url-pattern>

</web-resource-collection>

<user-data-constraint>

<description/>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>

</security-constraint>

de esta manera, todas las solicitudes que se realicen pasan a ser del tipo CONFIDENTIAL que en la práctica equivale a ser redirigidas al protocolo HTTPS

## struts.xml

Carpeta /WEB-INF/classes

Con el fin de poder proporcionar desde la primera pantalla un soporte idiomático, la visualización de esta primera pantalla será inicializada a través del framework struts. La acción que se inicia en index.html es validarLogin.action que es mapeada al método LoginAction.validarLogin():  
<action name="validarLogin" class="es.ubu.alu.mydatabasejc.actions.LoginAction" method="validarLogin">

<result name="error">/login.jsp</result>

<result type="redirectAction">

<param name="actionName">inicial</param>

<param name="namespace">/consulta</param>

<param name="parse">false</param>

</result>

</action>

Cuando, de resultas de la ejecución del método validarLogin, se devuelva “error”, se presentará de nuevo la pantalla login.jsp. Si, por el contrario, se devuelve “success” se deberá redirigir la acción a /consulta/inicial.action que presentará la información relativa a la conexión conseguida en la pantalla inicial de la aplicación.

## login.jsp

Carpeta /

La solicitud HTTP se deberá realizar mediante el protocolo POST evitando que los datos enviados sean incluidos en la url enviada. En el formulario se incluirán los tres parámetros indicados así como la acción que debe intentar realizar el login:  
<s:form action="login.action" method="POST" namespace="/">

<s:textfield name="url" key="URL"/>

<s:textfield name="usuario" key="Usuario"/>

<s:textfield name="password" key="Contrasena"/>

<s:submit key="Conectar"/>

</s:form>

El mensaje indicativo si existe algún problema aparecerá en la misma pantalla de identificación para que el usuario pueda hacer un nuevo intento de conexión. Para ello se dota al jsp de las líneas:

<s:actionerror/>

<s:actionmessage/>

que se transformarán en una lista html de mensajes cargados por el programa correspondiente

## package.properties

Paquete es.ubu.alu.mydatabasejc

Se puede apreciar bajo el atributo **key** los textos que pasados a través de los ficheros **package.properties** nos permiten la internacionalización de la pantalla:

# Etiquetas estáticas en pantalla

Contrasena=Contraseña

A modo de ejemplo, el fichero **package\_en.properties** daría soporte a navegadores configurados en el idioma inglés:

# Etiquetas estáticas en pantalla

Contrasena=Password

Usuario=User

Conectar=Connect

## struts.xml

La acción login.action es redirigida a LoginAction.login() que establecerá la conexión física con la base de datos en función de los datos suministrados en el formulario y añadirá a la sesión web del usuario el objeto que contiene la conexión como tal y los datos adicionales de la misma por si hay que reestablecerla en el futuro. En struts.xml el mapeo de la acción y el retorno del proceso:

<action name="login" class="es.ubu.alu.mydatabasejc.actions.LoginAction" method="login">

<result name="error">/login.jsp</result>

<result>/inicio.jsp</result>

</action>

## LoginAction.java

Paquete es.ubu.alu.mydatabasejc.actions

Esta clase es la encargada de recoger las solicitudes del usuario relativas a los distintos procesos de login e indicar a struts que debe hacer a continuación.

Los datos del formulario HTML son recogidos en las propiedades declaradas en la clase java con el mismo nombre que se definió en los propios campos del formulario. Las propiedades deben estar dotadas de sus métodos get y set oportunos:

private String url;

private String usuario;

private String password;

La respuesta HTTP obtenida presentará la información inicial de la base de datos si se consigue establecer una conexión válida o un mensaje indicativo del problema encontrado si no se consigue obtener una conexión válida.

Como requisitos de datos, la url y el nombre del usuario son datos obligatorios. Para controlar la validez de url y usuario, se utiliza el método validate de la correspondiente clase action de java.

Los métodos más importantes definidos en esta clase son:

* logear(): Se utiliza para que el usuario pueda ir directamente a la pantalla para hacer login, aunque ya se encuentre logado en una base de datos.
* login(): Recoge url, usuario y clave del formulario html e intenta realizar una conexión a la base de datos. Si tiene éxito, se guardará en la sesión del usuario el objeto que contiene la conexión a la base de datos.
* validarLogin(): Comprueba que el usuario disponga de una sesión. A continuación comprueba que el usuario disponga de una conexión a la base de datos ya establecida previamente y que sea válida. Si dispone de una conexión pero no es válida se intenta realizar una nueva conexión con los mismos datos previos. Si no ha tenido previamente una conexión, se presenta al usuario el formulario para que ingrese url, usuario y clave.
* validate(): Comprueba los valores del formulario html enviados. Se vuelve a la misma pantalla si alguno de los datos no es correctamente validado (url o usuario vacío). La siguiente línea, a modo de ejemplo, muestra cómo conseguir que se muestre un mensaje idiomatizado cuando se produce un fallo de validación:

if ("".equals(url)) addFieldError("url", "La.URL.es.un.dato.obligatoria");

* login(String url, String usuario, String password): Se trata de un método private, llamado desde el método login() y desde el método validarLogin(). Este método es el responsable de intentar establecer finalmente la conexión con la base de datos. Si finalmente tiene éxito, también añadirá la instancia de ConnectionImpl a la sesión del usuario con el nombre “conexion” para ser recuperada en llamadas posteriores.

## ConnectionImpl.java

Paquete es.ubu.alu.mydatabasejc.jdbc

El objetivo de esta clase es contener la conexión JDBC oportuna con la base de datos, junto con el resto de datos relevantes de la misma (url, usuario, clave).

También dispone de una serie de métodos para trabajar con la conexión JDBC contenida en ella. Los métodos más importantes son:

* ConnectionImpl(String url, String usuario, String password): Es el constructor de la clase. Inicializa el driver JDBC oportuno en función de la url recibida e instancia el objeto completando la conexión JDBC con la base de datos. Cualquier error es devuelto previo registro en el log de la aplicación.
* finalize(): Método sobreescrito que finaliza la conexión jdbc antes de que el objeto sea finalizado por el recolector de basura. De esta manera nos aseguramos que no queden conexiones con la base de datos abiertas innecesariamente
* isClosed(): true si la conexión con la base de datos está cerrada o no existe, false en caso contrario
* isValid(int timeout): true si la conexión con la base de datos es válida, false en caso contrario

## ConnectionException.java

Paquete es.ubu.alu.mydatabasejc.exceptions

Esta clase hereda de la clase Throwable y permite registrar en el log definido, no solamente los errores producidos en operaciones de establecimiento y validación de conexión a la base de datos, si no también adjuntar en el mismo log información relevante de los datos del contexto utilizados para la conexión.

Por ejemplo, si no se puede inicializar el driver JDBC, se acompaña cuál es el driver que intenta inicializarse. Si no se consigue una conexión, se acompaña al mensaje de error la url y el usuario que intentaron realizarla.