Acceso a la base de datos

Índice

1. Introducción

En esta sesión de integración procederemos a implementar algunas funcionalidades de los DAO's. También se deben crear los casos de prueba.

2. Creación de los DAO's

Vais a definir tres DAOs, uno por cada tabla. Para usarlos debéis definir una factoría de DAO. Esta factoría tendrá tres métodos que devolverán un objeto interfaz del DAO correspondiente. Dentro del paquete es.ua.jtech.proyint.dao se debe crear un paquete por cada DAO, donde dejaremos la interfaz correspondiente y su implementación JDBC.

Se describe brevemente el trabajo a realizar:

- DAO de usuario. Se implementarán los siguientes métodos:
  - public UsuarioTO selectUsuario(String login, String password): devuelve el usuario cuyo login y contraseña se pasa por parámetro.
  - public void addUsuario (UsuarioTO usuario): se añade el usuario en la BD.
  - public int delUsuario (UsuarioTO usuario): se elimina el usuario de la BD. Devuelve el número de registros modificados.
  - public ArrayList<UsuarioTO> getAllUsuarios (): devuelve todos los usuarios en la BD.
- DAO de libro. Se deben implementar los mismos métodos que en el DAO anterior.
- DAO de operación. Aquí el método a implementar es distinto:
  - public boolean realizaReserva (UsuarioTO usuario, LibroTO libro, Date finicio, Date ffin): inserta en la tabla operación la reserva del libro por parte del usuario indicado. También hay que cambiar el estado del usuario (modificando la tabla Usuario). Se debe realizar una transacción de las dos operaciones.

Todos los métodos deben lanzar una DAOException. Aquí tenéis un ejemplo de implementación:

```
public UsuarioTO selectUsuario(String login, String password) throws
DAOException {
        UsuarioTO usuario = null;

        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;
```

```
        try {
                String query = "select * from USUARIO where LOGIN = ?
and PASSWORD = ?"

                conn =
FactoriaFuenteDatos.getInstance().createConnection();
                st = conn.prepareStatement(query);
                st.setString(1, login);
                st.setString(2, password);
                rs = st.executeQuery();

                if (rs.next()) {
                        usuario = new UsuarioTO();
                        usuario.setLogin(login);
                        usuario.setPassword(password);
                        usuario.setNombre(rs.getString("nombre"));
usuario.setApellido1(rs.getString("apellido1"));
usuario.setApellido2(rs.getString("apellido2"));
usuario.setEstado(Enum.valueOf(EstadoUsuario.class,
rs.getString("estadoUsuario")));
usuario.setTipo(Enum.valueOf(TipoUsuario.class,
rs.getString("tipoUsuario")));
                }
        } catch (SQLException sqle) {
                throw new DAOException("Error en el select de
usuario", sqle);
        } finally {
                try {
                        if (rs != null) {
                                rs.close();
                                rs = null;
                        }
                        if (st != null) {
                                st.close();
                                st = null;
                        }
                        if (conn != null) {
                                conn.close();
                                conn = null;
                        }
                } catch (SQLException sqlError) {
                        throw new RuntimeException("Error cerrando las
conexiones", sqlError);
                }
        }

        return usuario;
}
```

Si os fijáis, usamos un PreparedStatement. Es para evitar el *SQL injection*.

3. Casos de prueba

También tenéis que definir los casos de prueba. Al trabajar con bases de datos la definición de los casos de prueba es un poco más complicada. Podéis lanzar los casos de prueba con la base de datos recién instalada. Para ello, hemos creado una tarea Ant que introduce los datos a usar.

Aquí tenéis un ejemplo de cómo se pueden hacer dos casos de prueba. El primero comprueba si se selecciona correctamente un usuario y el segundo si no se selecciona (el usuario no existe en la base de datos).

```
public void testSelectUsuario() throws DAOException {
        // El usuario/login miguel/cazorla debe existir en la BD
        UsuarioTO unUsuario = dao.selectUsuario("miguel","cazorla");

        Assert.assertNotNull("Usuario nulo", unUsuario);
        Assert.assertNotNull("Usuarion sin nombre",
unUsuario.getNombre());
}

public void testSelectNoUsuario() throws DAOException {
        // El usuario/login asdfsa/geraghss5tor no deberia existir en
la BD
        UsuarioTO unUsuario =
dao.selectUsuario("asdfsa","geraghss5tor");

        Assert.assertNull("Usuario no nulo, no deberia existir",
unUsuario);
}
```