**M06\_U4\_(BaseX-XQJ-XPATH/XQUERY)**

**Cosas a tener cuenta:**

Paquetes con los que trabaja: javax.xml.xquery.\*;

Objetos:

* **XQDataSource** 🡪 Crear la conexión.
* **XQConnection** 🡪 Objeto de conexión.
* **setProperty**("serverName", SERVER\_NAME) 🡪 Propiedades de la conexión.
* **setProperty**("port", PORT) 🡪 Propiedades de la conexión.
* **getConnection**(USER, PASSWORD) 🡪 Obtenemos la conexión a través de su usuario y contraseña.
* **XQExpression** 🡪 Objeto para crear consultas (inserción, obtención, etc…)
* **XQResultSequence** 🡪 Es un ResultSet. Obtiene los datos de una consulta.
* **executeQuery**(“Expresión XPath-XQuery”);

Sintaxis:

* Parar acceder a la “colección” hay que indicar el nombre de la misma y el nombre del fichero “xml” seperado por una barra “/”:
  + doc(“canal”/canals.xml”)/collection(“canals”)//Canals/canal…
  + doc(“canals”)//Canals/canal

**Abrir conexión**:

Parar abrir conexión necesitamos una serie de parámetros:

1. **Nombre del servidor**: private final String SERVER\_NAME="localhost";
2. **Usuario**: private final String USER="admin";
3. **Contraseña**: PASSWORD="admin";
4. **Puerto de conexión**: PORT="1984";
5. **XQConnection**: private XQConnection con=null;
6. **XQDataSource**: fábrica para obtener objetos XQConnection

Creamos un método para hacer uso cuando lo necesitemos:

private void obrir() throws GestorException, XQException{

XQDataSource xqs = new BaseXXQDataSource();

xqs.setProperty("serverName", SERVER\_NAME);

xqs.setProperty("port", PORT);

con = xqs.getConnection(USER, PASSWORD);

gestor = new GestorCanal(con);

}

**Insertar:**

Para insertar un Canal, debemos utilizar la instrucción (insert node):

**insert node <nodo>Valor</nodo> into doc(“Canals/canals.xml”)/collection(“canals”)//canals**

Veamos un ejemplo con un método que inserta canales:

public void inserir(Canal nouCanal) throws GestorException, XQException {

//TODO

if(obtenirCanal(nouCanal.getId()) != null) {

throw new GestorException(CLAU\_DUPLICADA);

}

try {

String insert = "insert node " + formarXML(nouCanal) + " into " + ARREL + "canals";

XQExpression exp = con.createExpression();

exp.executeQuery(insert);

exp.close();

} catch (XQException e) {

throw new GestorException(e.getMessage());

}

}

Este método recibe un objeto de tipo Canal por parámetro, comprueba que sea “null” (de lo contrario lanza una exepción). Si no es “null”, inserta el canal transformándolo en un XML con la ayuda del método formarXML(Objeto) que veremos definido más abajo. Utiliza un objeto “XQExpression” para indicar que vamos a crear una nueva expresión o query, después con el método “executeQuery(“query”)”, ejecutamos la consulta. Para finalizar, cerramos la expresión.

**Actualizar:**

La actualización no la hemos trabajado en el ejercicio, pero veremos un ejemplo. Para actualizar un nodo, debemos recorrerlos uno a uno hasta encontrar el que cumpla con la condición que le indiquemos:

Recorremos: for **$i** in /bailes/baile[**@id="2"**] **🡪** Recorrerá los nodos hasta encontrar el que tenga el id =”2”.

Indicamos la acción: return replace node **$i** 🡪 Indicamos que “reemplace” lo que tenemos almacenado en “$i”.

Indicamos porque lo tiene que cambiar: with <prueba id=”4”>ValorPrueba</prueba>

Ejemplo completo:

for $i in /bailes/baile[@id="2"] return replace node $i with <prueba id="1">Prueba</prueba>

**Eliminar:**

Para eliminar un Canal, es el mismo proceso que para insertar, sólo que en vez de comprobar que el nodo no exista, tenemos que comprobar que efectivamente si existe. En caso contrario, lanzará una Excepción. También hay que recalcar que hay que utilizar la expresión “delete node”:

Expresión completa:

delete node doc(“canals/canals.xml”)/collection(“canals”)//canals/canal[id=”numID”]

Ejemplo método:

public void eliminar(int idCanal) throws GestorException {

if(obtenirCanal(idCanal) == null) {

throw new GestorException(CLAU\_INEXISTENT);

}

try {

String elim = "delete node " + ARREL + "canals/canal[id=\"" + idCanal + "\"]";

XQExpression exp = con.createExpression();

exp.executeQuery(elim);

exp.close();

} catch (XQException e) {

throw new GestorException(e.getMessage());

}

}

Este método, recibe un parámetro de tipo entero el cual representa el número de id del canal. Comprueba que el canal no exista (si no lanzará un excepción). Si no existe, ejecuta la instrucción de la variable String “elim” por medio del objeto XQExpression haciendo uso de su método “executeQuery”. Por último, cerramos la expresión.

**Obtener:**

Podemos obtener un canal o todos pasándole un id para el primero caso y un valor o filtro en el segundo. Vemos ambos casos:

Obtener canal por ID:

El primer paso es crear la expresión que después pasaremos como parámetro al método “executeQuery” del objeto “XQExpression”:

String query = ARREL + "canals/canal[id=\"" + idCanal + "\"]";

El segundo paso es crear la expresión:

XQExpression exp = con.createExpression();

El tercer paso es guardar el resultado del objeto XQExpression en un objeto “XQResultSequence”:

XQResultSequence result = exp.executeQuery(query);

El cuarto paso es coger la instancia del objeto “XQResultSequence 🡪 result” y ver si hay algo en él:

if(result.next()) {

El quinto paso es almacenar ese resultado en un objeto de tipo Canal:

canal = obtenCanal(result.getItemAsString(null));

**NOTA**: la instancia de XQResultSequence (result), tiene el método “getItemAsString(null)” que devuelve un String en formato XML con todos los datos del nodo.

Vemos el método completo:

public Canal obtenirCanal(int idCanal) throws GestorException {

Canal canal = null;

String query = ARREL + "canals/canal[id=\"" + idCanal + "\"]";

try {

XQExpression exp = con.createExpression();

XQResultSequence result = exp.executeQuery(query);

if(result.next()) {

canal = /\*Utilitats.obteCanal\*/ obtenCanal(result.getItemAsString(null));

}

exp.close();

} catch (XQException e) {

throw new GestorException(e.getMessage());

}

return canal;

}

Obtener Lista de canales filtrando por subscriptor:

1. El primer paso, creamos una lista de objetos de tipo Canal: List<Canal> canals = new ArrayList();
2. Segundo paso, creamos la expresión: String query = "for $i in " + ARREL + "canals/canal where $i/subscriptors[subscriptor=\"" + subs + "\"] return $i";
3. Tercer paso instanciamos el objeto “XQExpression”: XQExpression **exp** = con.createExpression();
4. Cuarto paso, instanciamos el objeto “XQResultSequence”: XQResultSequence **result** = exp.executeQuery(query);
5. Recoremos el “result”: while(**result**.next()) {
6. Añadimos a la lista “canals” los nodos convertidos en objetos Canal que cumplan con el filtro expresado en el punto “2”: canals.addobtenCanal(**result**.getItemAsString(null)));
7. Cerramos la expresión: **exp**.close();

El método recibe por parámetro un String con el nombre del subscriptor que actuará de filtro. Después recoremos cada nodo para ver si cumple las condiciones del filtro. El resultado lo recorremos con “while” y vamos añadiendo ese resultado a la lista de canales.

Vemos el método al completo:

public List<Canal> obtenirCanalPerSubscriptor(String subs) throws GestorException {

List<Canal> canals = new ArrayList();

String query = "for $i in "+ARREL+"canals/canal where $i/subscriptors[subscriptor=\""+subs+"\"] return $i";

try {

XQExpression exp = con.createExpression();

XQResultSequence result = exp.executeQuery(query);

while(result.next()) {

canals.add(/\*Utilitats.obteCanal\*/obtenCanal(result.getItemAsString(null)));

}

exp.close();

} catch (XQException e) {

throw new GestorException(e.getMessage());

}

return canals;

}

**Métodos de ayuda para formar un XML y obtener un Objeto Canal:**

**Método formarXML**: este método recibe un objeto Canal y lo convierte en un String con formato XML.

private String formarXML(Canal canal) {

StringWriter sw = new StringWriter();

try {

context = JAXBContext.newInstance(Canal.class);

marshall = context.createMarshaller();

marshall.setProperty(Marshaller.JAXB\_FRAGMENT, true);

marshall.marshal(canal, sw);

} catch (JAXBException e) {

System.out.println(e.getMessage());

}

return sw.toString();

}

**Método obtenCanal**: este método recibe una cadena y la transforma en un objeto de tipo Canal.

private Canal obtenCanal(String cadena) {

Canal canal = null;

StringReader reader = new StringReader(cadena);

try {

context = JAXBContext.newInstance(Canal.class);

unMarshall = context.createUnmarshaller();

canal = (Canal) unMarshall.unmarshal(reader);

} catch (JAXBException e) {

System.out.println(e.getMessage());

}

return canal;

}