Acceso a eXist desde Java

API XQJ

Introducción

- Ya hemos visto acceso a ficheros XML (DOM y SAX) , ahora veremos una API para acceder a la bbdd eXist
- Las APIs más conocidas son:
  - API XML:DB → cuyo objetivo es la definición de un método común de acceso a SGBD XML. Permite consulta, creación y modificación de contenido. La última actualización fue en el 2001 y aunque se sigue utilizando bastante se puede considerar obsoleta
  - API XQJ → propuesta de estandarización de interfaz Java para el acceso a bbdd XML nativas basado en modelo de datos XQuery. Es similar a JBDC. Es independiente del fabricante, fácil de usar pero solo permite realizar consultas.

XQJ

- Para descargar la API accederemos a http://xqj.net/exist/
- Para trabajar con la API necesitamos los siguientes import:

```
import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQPreparedExpression;
import javax.xml.xquery.XQResultItem;
import javax.xml.xquery.XQResultSequence;
import net.xqj.exist.ExistXQDataSource;
```

Configurar una conexión

- **XQDataSource:** identifica la fuente física de datos a partir de la cual crear conexiones. Cada implementación definirá las propiedades necesarias para efectuar la conexión.

```
XQDataSource server = new ExistXQDataSource();
server.setProperty (“serverName”,”localhost”);
server.setProperty (“port”,”8080”);
server.setProperty (“user”,”admin”);
server.setProperty (“password”, “austria”);
```

- **XQConnection:** representa una sesión con la bbdd, manteniendo información de estado, transacciones, expresiones ejecutadas y resultados.

```
XQConnection conn = server.getConnection();
```

Clases y métodos para procesar consultas

- **XQExpression:** objeto creado a partir de una conexión para la ejecución de una expresión. Devuelve un **XQResultsetSequence.** La ejecución se produce llamando al método **executeQuery.**
- **XQPreparedExpression:** objeto creado a partir de una conexión para la ejecución de una expresión múltiples veces.
- **XQDynamicContext:** representa el contexto dinámico de una expresión (zona horaria, variables a usar en la expresión)
- **XQStaticContext:** representa el contexto estático para la ejecución de expresiones

- **XQItem:** representación de un elemento **XQuery**. Es inmutable y una vez creado su estado interno no cambia.
- **XQResultItem:** representación de un elemento de un resultado.
- Con XQJ no se necesita seleccionar la colección de los documentos XML, la búsqueda la realiza en todas las colecciones. Por tanto, a la hora de hacer consultas indicaremos la colección o el documento de la colección

Ejemplo

```
import javax.xml.xquery.*;
import net.xqj.exist.ExistXQDataSource;
public class verproductos {
	public static void main(String[] args){
		try{
		XQDataSource server = new ExistXQDataSource();
		server.setProperty ("serverName", "localhost");
		server.setProperty ("port","8080");
		server.setProperty ("user","admin");
		server.setProperty ("password","austria");
		XQConnection conn = server.getConnection();
		XQPreparedExpression consulta;
		XQResultSequence resultado;
		consulta = conn.prepareExpression ("for $pr in doc('nueva/productos.xml')/productos/produc return $pr");
		resultado = consulta.executeQuery();
		while (resultado.next()) {
			System.out.println(resultado.getItemAsString(null));
		}
		conn.close();
		} catch (XQException ex) {System.out.println("Error al operar"+ex.getMessage());}
	}
}
```

Problemas XQJ_1

1. Descarga la API y prueba el ejemplo anterior
2. Accede vía navegador local a la documentación de la API recién descargada
3. Realiza un programa que devuelva el número de productos con precio mayor a 50.
4. Realiza un programa que devuelva todos los empleados del departamento 10.

Introducción de parámetros en consultas XQuery

- Para introducir parámetros en una consulta el proceso utilizamos el método prepareExpression. Lo vemos con un ejemplo:

```
/* Previamente vendría la conexión a la base de datos configurando XQDataSource
   A continuación definimos la variable $x que representa el ID de un departamento */
                consulta = conn.prepareExpression (
                                "declare variable $x as xs:int external;"+
                                "/EMPLEADOS/EMP_ROW[DEPT_NO=$x]");
/* Introducimos el valor del parámetro de la consulta */
               int valor=10;
                consulta.bindInt(new QName("x"), valor, null);
/* Más opciones en aquí */
                resultado = consulta.executeQuery();
                while (resultado.next()) {
                        System.out.println("Element E2: "+
                                        resultado.getItemAsString(null));
                }
```

Crear un nuevo archivo XML con datos del eXist

```
/* Ejecutamos la siguiente consulta: “Devuelve todos los datos de los empleados del departamento 10 pero organizados según la estructura de los datos en el documento empleados.xml” */
                        consulta = conn.prepareExpression(
                                        "let $titulo:= /EMPLEADOS/TITULO return " +
                                        "<EMPLEADOS>{$titulo} "+
                                        "{for $em in /EMPLEADOS/EMP_ROW[DEPT_NO=10] "+
                                        "return $em}</EMPLEADOS>");
                        resultat = consulta.executeQuery();
/* Escribimos los datos de la consulta en un fichero xml */
                        writer = new BufferedWriter(new FileWriter(fitxer));
/* Cabecera XML */
                        writer.write("<?xml version='1.0' encoding='UTF-8'?>");
                        writer.newLine();
/* Nodos XML */
                        while (resultat.next()) {
                                String cad = resultat.getItemAsString(null);
                                writer.write(cad);
                                writer.newLine();
                        }
```

Problemas XQJ_2

1. A partir del documento *universidad.xml*, haz un programa que muestre los empleados del departamento cuyo tipo es elegido por el usuario. Si no hay empleados o el tipo de departamento aportado por el usuario no existe, se debe de informar al usuario.
2. A partir de los documentos *productos.xml* y *zonas.xml*, haz un programa que reciba un número de zona por parámetro y genere un documento con nombre *zonaXX.xml* donde XX es la zona solicitada. El documento debe contener los productos de esta zona y las siguientes etiquetas: <cod_prod>, <denominación>, <precio>, <nombre_zona>, <director> y <stock>. Donde el *stock* se calcula restando el *stock* actual y el stock mínimo.