HELPER DOCUMENT | MANEJO BBDD XML

INFORMACION GENERAL

este es un documento de ayuda para saber manejar una bbdd no relacional de xml, vamos a necesitar una bbdd xml como puede ser existdb. el documento explicativo se se divide en:

- ***primero vamos a ver consultos y metodos dentro del propio IDE web.***
- ***como podemos utilizar todo esto en un proyecto de java.***

BBDD XML | CONSULTAS & METODOS

CONSULTAS BASICAS

toda la informacion sobre consultas basicas la tendras a continuacion, desde lo mas basico a lo mas complejo:

- **consultas basicas → elementos raiz y padre**

<!-- mostrar etiquetas de los documentos que cuelgan del nodo raiz -->

/

<!-- mostrar etiquetas de los documentos que cuelgan del nodo departamentos -->

/departamentos

<!-- mostrar etiquetas de los documentos que cuelgan del nodo departamentos/dep_row -->

/departamentos/dep_row

<!-- mostrar etiquetas de los documentos que cuelgan del nodo departamentos/dep_row pero no devuelve el nombre de dept_row -->

/departamentos/dep_row/node()

<!-- motrar el nombre de los departamentos, pero solo en texto por la funcion text() -->

/departamentos/dep_row/nombre/text()

<!-- mostrar todos los nodos "x" independientemente del fichero -->

//loc

- **consultas basicas → operadores**

<!-- mostrar los nodos que esten a 1 nivel de profundidad de la raiz -->

/*/dept_no

<!-- mostrar los nodos que esten a 2 nivel de profundidad de la raiz -->

/*/*/dept_no

<!-- mostrar los nodos que van dentro de x y los subnodos -->

/departamentos/*

<!-- depende del contexto muestra todas las etiquetas de los documentos de la coleccion -->

/*

- **consultas basicas → condiciones de seleccion**

<!-- operadores para las condiciones de seleccion -->

[] → selecciona elementos concretos

<, >, <=, >=, =, != → operadores simples

or, and y not → operadores para concatenar condiciones

| → unir varias rutas

<!-- mostrar todos los nodos dentro de emp_row que el dept_no sea 10 -->

/empleados/emp_row[dept_no = 10]

<!-- mostrar los apellidos y el dept_no -->

/empleados/emp_row/apellido|/empleados/emp_row/dept_no

<!-- mostrar los apellidos de los empleados que dept_no sea 10 -->

/empleados/emp_row[dept_no = 10]/apellido

<!-- mostrar los nodos de los empleados que dept_no NO sea 10 -->

/empleados/emp_row[not(dept_no = 10)]

<!-- mostrar los apellidos y el oficio que tengan el dept_no = 10 -->

/empleados/emp_row[dept_no = 10]/apellido | /empleados/emp_row[dept_no]/oficio

<!-- mostrar los apellidos de los empleados con dept_no = 10 y cobren mas de 1300 euros -->

/empleados/emp_row[salario > 1300 and dept_no = 10]/apellido

- **consultas basicas → funciones**
    - **[numero] →** devuelve el elemento que ocupa el lugar indicado
    - **last() →** selecciona el utlimo elemento del conjunto seleccionado
    - **count() →** cuenta el numero de elementos seleccionados
    - **sum() →** devuelve la suma del elemento seleccionado
        - si la etiqueta al sumar se considera como string, habra que convertirlo a un numero → number(valor)
    - **max(), min(), avg() →** devuelve el valor maximo, minimo y la media de los elementos seleccionados
    - **name() →** devuelve el nombre del elemento seleccionado
    - **concat(cad1, cad2, ...) →** concatena las cadenas introducidas
    - **starts-with(cad1, cad2, ...) →** obtiene los elementos donde la *cadena1* empieza por la *cadena2*
    - **contains(cad1, cad2) →** obtiene los elementos en los cuales la *cadena1* contiene a la *cadena2*
    - **string-length(argumento) →** devuelve el numero de caracteres
    - *puedes encontrar otras funciones en el siguiente enlace:* pincha aqui

---

ATRIBUTOS

no se consideran nodos hijos, sino etiquetas añadidas al nodo, nos referimos a ellos a traveas del @

<!-- mostrar los datos de los departamentos que el tipo sea A -->

/universidad/departamento[@tipo="A"]

<!-- mostrar los datos de los empleados que tengan un salario > 2100 -->

/universidad/departamento/empleado[@salario>"2100"]

ACTUALIZACIONES EN EXISTS

vamos a diferenciar entre los metodos insertar, modificar y eliminar elementos de los documentos, piensa que podemos utilizar los filtros y las condiciones que hemos visto en el apartado de consultas.

- **actualizaciones → insert** update insert ELEMENTO into EXPRESION XPATH

<!-- ejemplo de insertar una zona en la ultima opcion -->

update insert

<zona>

<cod_zona>50</cod_zona>

<nombre>Madrid-OESTE</nombre>

<director>AliciaRamos Martin</director>

</zona>

into /zonas

<!-- ejemplo de insertar una cuenta en el documento sucursales.xml del tipo PENSIONES a la sucursal SUC1 -->

update insert

<cuenta tipo=“PENSIONES”>

<nombre>Alberto Morales</nombre>

<numero>30302900</numero>

<aportacion>5000</aportacion>

</cuenta>

into /sucursales/sucursal[@codigo=“SUC1”]

- **actualizaciones → replace** update replace NODO with VALOR_NUEVO

<!-- Cambia la etiqueta director de la zona 50 y su contenido, en el documento zonas.xml -->

update replace

/zonas/zona[cod_zona=50]/director

with

<directora>Pilar Martín Ramos</directora>

<!-- Cambia el nodo completo DEP_ROW del departamento 10, por los nuevos datos y las etiquetas que escribamos. -->

update replace

/departamentos/DEP_ROW[DEPT_NO=10]

with

<DEP_ROW>

<DEPT_NO>10</DEPT_NO>

<DNOMBRE>NUEVO10</DNOMBRE>

<LOC>TALAVERA</LOC>

</DEP_ROW>

- **actualizaciones → value** update value NODO with 'VALOR_NUEVO'

<!-- Cambia el apellido del empleado 7369, del documento empleados.xml -->

update value

/EMPLEADOS/EMP_ROW[EMP_NO=7369]/APELLIDO

with

"Alberto Montes Ramos"

<!-- Cambia el atributo tipo de la primera cuenta de la sucursal SUC3, del documento sucursales.xml -->

update value

/sucursales/sucursal[@codigo="SUC3"]/cuenta[1]/@tipo

with

"NUEVOTIPO"

- **actualizaciones → delete** update delete expresión xpath'

<!-- Elimina la zona con código 50, en el documento zonas.xml -->

update delete

/zonas/zona[cod_zona=50]

- **actualizaciones → rename** update rename NODO as NUEVO_NOMBRE'

```
<!-- Cambia de nombre el nodo EMP_ROW del documento empleados.xml -->
update rename
        /EMPLEADOS/EMP_ROW
as
        'fila_emple'
```

BBDD XML | PROYECTO JAVA QXAPI

CONEXION Y DESCONEXION A LA BBDD

metodos para abrir y cerrar la conexion a la bbdd.

```
private XQDataSource server;
private XQConnection connection;

public void iniciarConexion () {
        try {
                XQDataSource server = new ExistXQDataSource();

                server.setProperty(“serverName”, “localhost”);
                server.setProperty(“port”, “8083”);
                server.setProperty(“user”, “admin”);
                server.setProperty(“password”, “practicasiap”);

                connection = server.getConnection();
        } catch (XQException ex) {
                System.out.println(" excepcion de xqexception");
        }
}

public void cerrarConexion () {
```

```
        try {
                connection.close();
        } catch (XQException ex) {
                System.out.println(" excepcion de xqexception");
        }
}
```

EJECUTAR CONSULTAS EN EXISTS

metodo de ejemplo para saber crear e imprimir consultas de la bbdd.

```
public void consultaSimple (String inputTextoConsulta) {
        XQPreparedExpression consulta;
        XQResultSequence resultado;

        consulta = con.prepareExpression(inputTextoConsulta);
        resultado = consulta.executeQuery();

        XQResultItem r_item;
        while (resultado.next()) {
                r_item = (XQResultItem) resultado.getItem();
                System.out.println("Elemento: " + r_ítem.getItemAsString(null));
        }
}

public static void main (String[] args) {
        OperacionesBBDD oper = new OperacionesBBDD()
        oper.abrirConexion()
        oper.consultaSimple()
        oper.cerrarConexion()
}
```

EJECUTAR COMANDOS EN EXISTS

metodo para ejecutar metodos a la bbdd, los metodos de insertar, modificar y eliminar elementos.

- **ejecutar comandos → insertar**

```
public void modificacion (String inputTextoConsulta) {
   // "update rename " + "/EMPLEADOS/fila_emple2 " + "as " + "'fila_emple'"
  try {
   XQExpression xqConsulta;
   xqConsulta = connection.createExpression();

   xqConsulta.executeCommand(inputTextoConsulta);
  } catch (XQException ex) {
   Logger.getLogger(OperacionesBBDD.class.getName()).log(Level.SEVERE, null, ex);
  }
}

public static void main (String[] args) {
        OperacionesBBDD oper = new OperacionesBBDD()
        oper.abrirConexion()
        oper.modificacion()
        oper.cerrarConexion()
}
```

- **ejecutar comandos → modificacion**

```
public void modificacion (String inputTextoConsulta) {
   // "update rename " + "/EMPLEADOS/fila_emple2 " + "as " + "'fila_emple'"
  try {
   XQExpression xqConsulta;
   xqConsulta = connection.createExpression();

   xqConsulta.executeCommand(inputTextoConsulta);
```

```
  } catch (XQException ex) {
    Logger.getLogger(OperacionesBBDD.class.getName()).log(Level.SEVERE, null, ex);
  }
}

public static void main (String[] args) {
        OperacionesBBDD oper = new OperacionesBBDD()
        oper.abrirConexion()
        oper.modificacion()
        oper.cerrarConexion()
}
```

- **ejecutar comandos → eliminar**

```
public void eliminacion (String inputTextoConsulta) {
    // "update delete " + "/EMPLEADOS/fila_emple2"
  try {
    XQExpression xqConsulta;
    xqConsulta = connection.createExpression();

    xqConsulta.executeCommand(inputTextoConsulta);
  } catch (XQException ex) {
    Logger.getLogger(OperacionesBBDD.class.getName()).log(Level.SEVERE, null, ex);
  }
}

public static void main (String[] args) {
        OperacionesBBDD oper = new OperacionesBBDD()
        oper.abrirConexion()
        oper.eliminacion()
        oper.cerrarConexion()
}
```