6 QIZX

6.1 INTRODUCCIÓN

Qizx es un motor de base de datos "XML Query (XQuery)" diseñado para ser embebido en una aplicación Java – típicamente como un "Servlet". Como tal, es principalmente usado como una clase de la API.

Para ayudar en las peticiones Xquery y desarrollo de base de datos, Qizx viene con dos herramientas que hacen fácil construir una base de datos, llenarla de documentos XML y realizar peticiones en la base de datos sin programar (excepto en XML Query).

Qizx Studio

Herramienta gráfica compuesta de un explorador para navegar sobre el contenido de la base de datos, más un área (XML Query workbench), donde se puede escribir y ejecutar peticiones Xquery y ver los resultados.

qizx

Herramienta de línea de comandos, la cual puede ser usada para crear y mantener la base de datos XML, o simplemente ejecutar peticiones XQuery.

En Qizx, una base de datos es llamada Biblioteca XML. Físicamente una biblioteca es almacenada en un directorio del disco. No existe límite en el número de bibliotecas que pueden ser creadas con Qizx.

El motor Qizx puede de hecho manejar varias bibliotecas (Libraries) al mismo tiempo. Esto permite una mejora en el reparto de recursos en el caso de que una aplicación necesite manejar varias bibliotecas.

Un grupo de bibliotecas (Library Group) es simplemente un conjunto de bibliotecas (Library) dentro de un directorio padre. Un grupo de bibliotecas puede ser abierto o creado en una simple operación por el motor Qizx.

Una biblioteca es normalmente parte de un Grupo de Bibliotecas y una biblioteca puede ser abierta independientemente y puede aún pertenecer a varios grupos.

En la práctica, se usará sólo una biblioteca. Es raro usar dos o más bibliotecas a menos que se quiera separar conjuntos de datos para algunas aplicaciones. Indexar temas puede ser una razón también.

6.2 QIZX STUDIO

6.2.1 CREAR UNA BIBLIOTECA USANDO QIZX STUDIO

Arrancar Qizx Studio

En Windows, en el directorio bin dentro de la distribución Qizx distribution contiene un ejecutable qizxstudio.exe (ó qizxstudio.bat), que puede ser iniciado directamente haciendo doble clic sobre el.

En Linux o Mac OS X u otro Unix, un shell script bin/qizxstudio puede ser iniciado de una consola windows o de un explorer gráfico.

Tras iniciar Qizx Studio se vería una ventana similar a la siguiente:

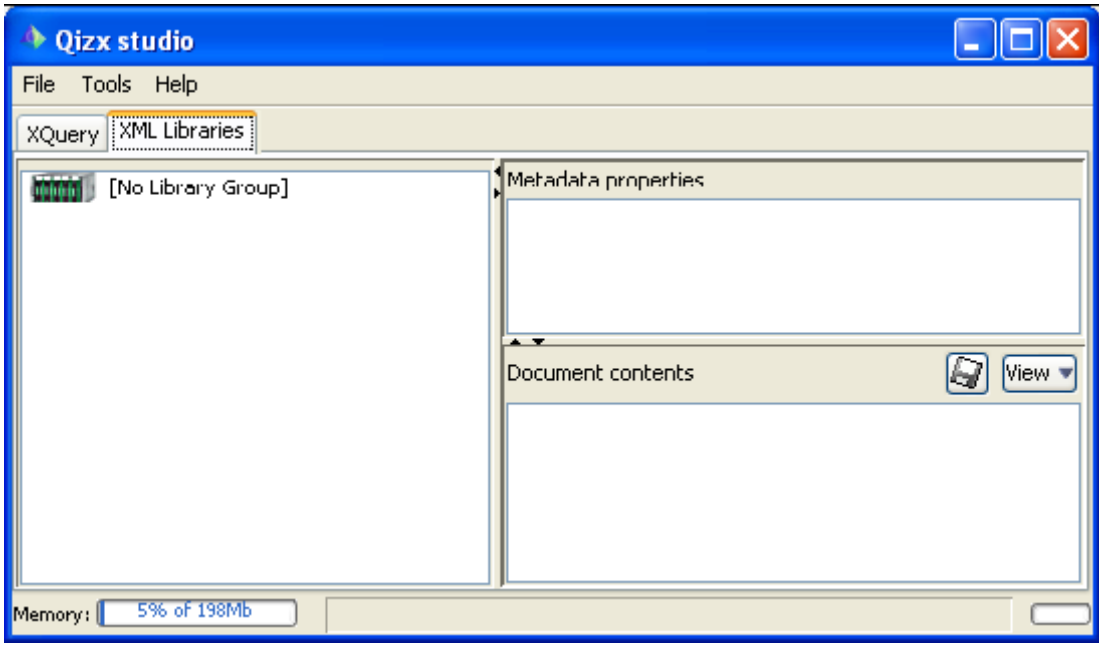

Figura 6-1: Arranque inicial de Qizx Studio

Notas:

• Hay dos etiquetas en Qizx Studio: "XQuery" para introducir y ejecutar peticiones XQuery, "XML Libraries" para navegar y modificar bibliotecas XML.

• La cabecera [No Library Group] significa que Qizx Studio no ha abierto todavía ningún grupo de bibliotecas. Todavía, es posible ejecutar peticiones XQuery sin acceso a alguna biblioteca.

Creación de una biblioteca (Library)

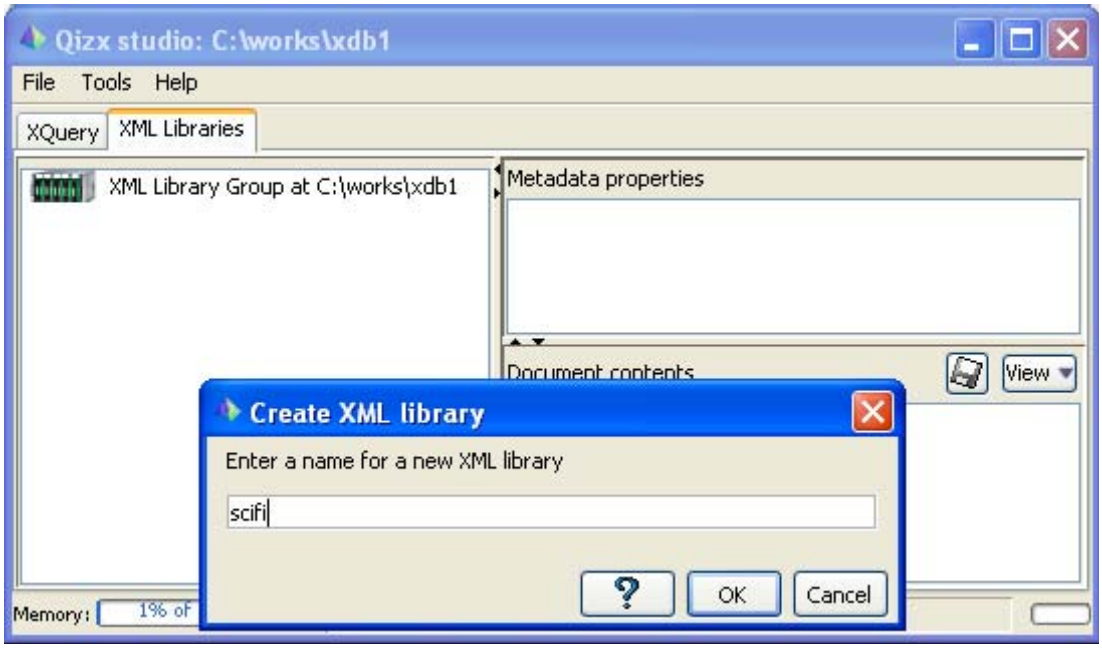

Figura 6-2: Creación de una biblioteca (Library)

1. Haciendo clic sobre el botón derecho del icono “Library Group” y eligiendo “Create Library Group” nos muestra una ventana de diálogo para seleccionar el directorio (nuevo o vacío). Se asume aquí “C:\works\xdb1” (aunque puede ser cualquiera).

2. Luego, la ventana de diálogo de arriba nos pregunta el nombre de la primera biblioteca dentro del grupo. Se asume aquí que el nombre elegido es “scifi”.

3. Cuando la biblioteca es creada, contiene la colección raiz, cuyo path es “/”. Haciendo clic en la colección raíz, se verán sus propiedades Metadata que por defecto aparecen en el lado derecho.

4. Es posible crear más bibliotecas con el menú que aparece al hacer un clic con el botón derecho en el icono del “Library Group”.

5. Para abrir un grupo existente de bibliotecas (Library Group) se usa la entrada de menú “Open XML Library Group” y se elige su directorio. También se puede directamente elegir una biblioteca (Library) en lugar de un grupo, pero en este caso, se puede controlar solamente una única biblioteca.

Nota: Una biblioteca puede ser abierta una sola vez por el motor de Qizx. Si se intenta abrir varias veces, se obtiene un mensaje de error indicando que la biblioteca está bloqueada.

Creación de una Colección (Collection)

Pulsando sobre el botón derecho del icono de una colección raíz, y eligiendo “Create sub-collection”, te pregunta por el nombre de una “Collection” (el nombre no

debe contener espacios en blanco. La colección es creada como “hijo” de la colección raíz. Si se elige el nombre “books”, el path de la colección es “/books”.

6.2.2 LLENAR UNA BIBLIOTECA DE COLECCIONES Y DOCUMENTOS

Importar documentos usando Qizx Studio

Asumiendo que se ha creado una biblioteca “scifi” como se explicó antes:

1. Se hace clic con el botón de derecho sobre el icono de la root Collection (path ‘/’) y se elige “Import Documents”. Un cuadro de diálogo aparece.

2. La operación de importar es realizada en dos pasos:

- Los ficheros y directorios son seleccionados a una lista de importación usando el botón “Add File/Folder”. El filtro “combo-box” permite filtrar los ficheros con la extensión de interés (generalmente .xml). Aquí se seleccionará el directorio entero “docs\samples\book_data” ó “docs/samples/book_data” dentro de la distribución Qizx. Como se usa un filtro *.xml sólo los ficheros con extensión .xml serán seleccionados.

  Después de la selección, el número de documentos seleccionados y su tamaño total en bytes son mostrados en la tabla.

  Esta operación de selección puede ser repetida en directorios o en un solo fichero XML. Los botones de “Remove” y “Clear all” permite editar la lista.

- Pulsando el botón “Start Import” se empezará la transacción de importar.

  Después de completarse la acción, el diálogo se puede cerrar con el botón “Close”. Errores de análisis podrían ser mostrados en la ventana de mensajes del diálogo. La velocidad de importación puede alcanzar los 2 Mb por segundo en un procesador de 3Ghz para grandes documentos, pero en un gran número de pequeños documentos puede proporcionalmente ralentizar este proceso.

Una vez que la importación es terminada, se debería ver lo siguiente:

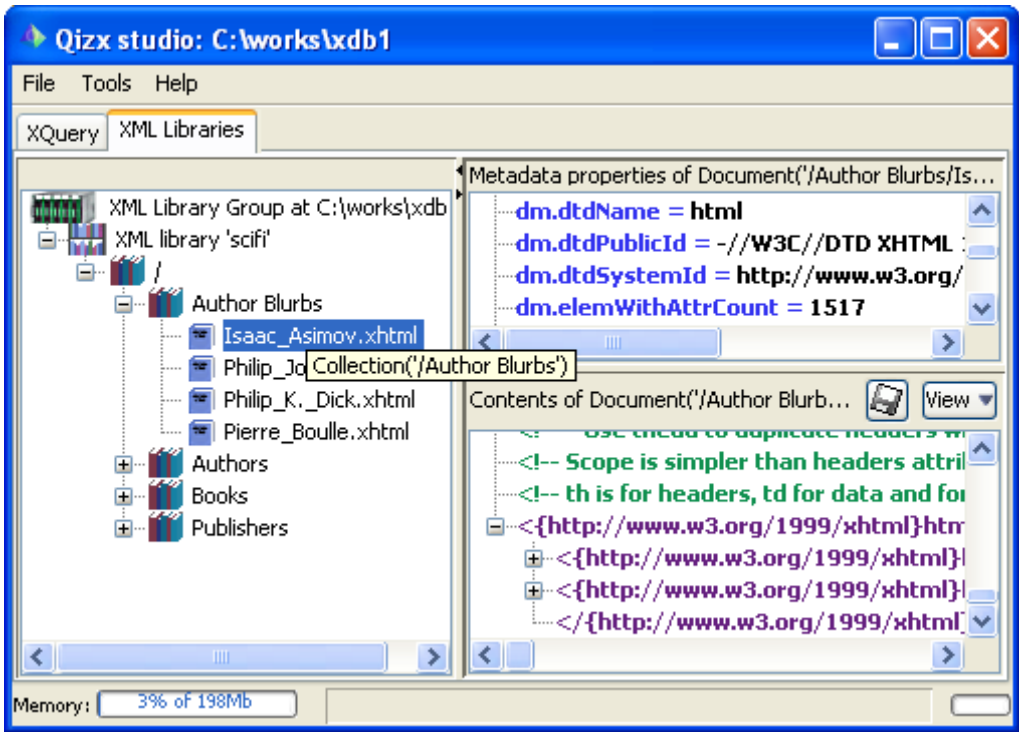

Figura 6-3: Estado de Qizx Studio después de importar los documentos

6.2.3 PETICIONES XQUERY SOBRE UNA BIBLIOTECA

En esta sección se va a ejecutar peticiones sobre la base de datos que se acaba de crear. Qizx Studio actualmente proporciona un básico entorno para editar y ejecutar peticiones XQuery.

Se prueba con la siguiente petición XQuery (fichero.xql):

```
 (: Find all books written by French authors. :)

declare namespace t = "http://www.qizx.com/namespace/Tutorial";

for $a in collection("/Authors")//t:author[@nationality =
"France"]

for $b in collection("/Books")//t:book[.//t:author =
$a/t:fullName]

return

$b/t:title
```

En Qizx Studio, pulsando sobre la etiqueta XQuery y luego usando el menú File-> Open XQuery se cargará el fichero mencionado arriba. Luego se pulsará el botón Execute para ejecutar la petición. Después de la ejecución, se debería obtener algo similar a lo siguiente:

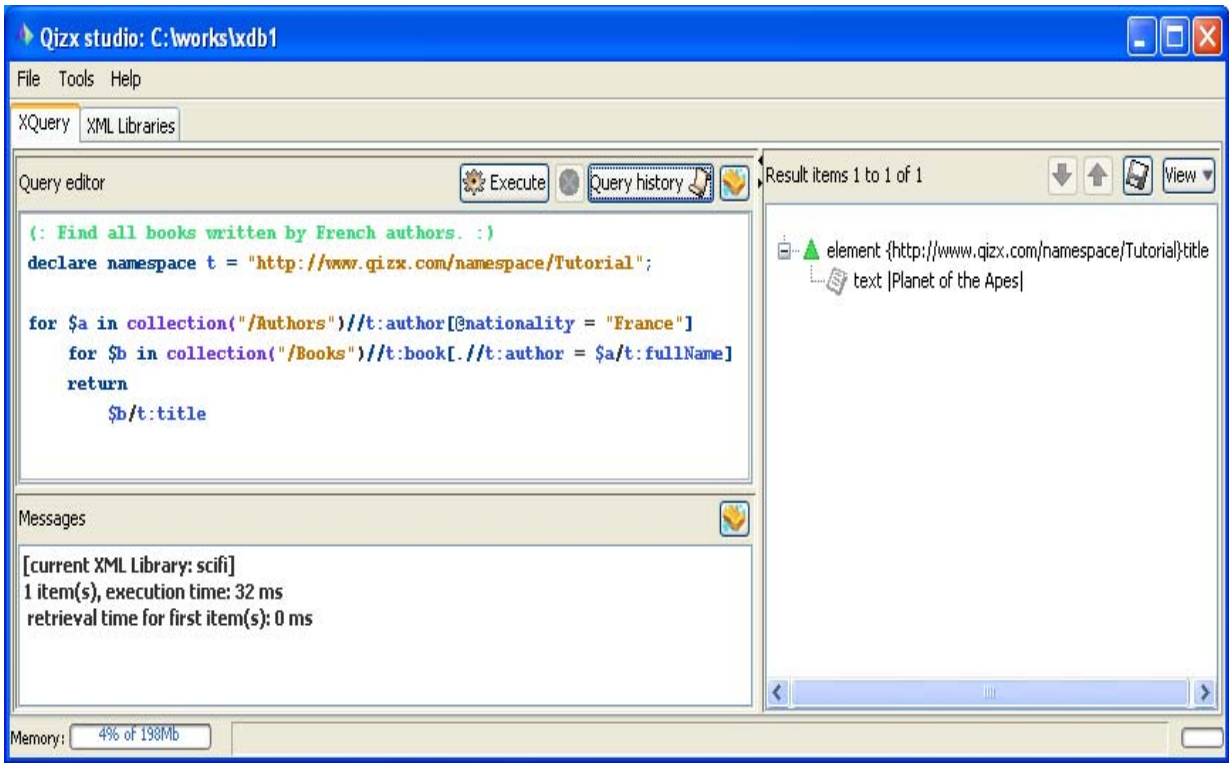

Figura 6-4: Resultado de una petición XQuery

Nota: También se puede guardar un fichero o una petición que se haya introducido o editado con Qizx Studio.

El resultado de la búsqueda contiene un artículo, el cual es un elemento t:titles cuyo valor string es “Planet of the Apes”.

Se puede por ejemplo, cambiar el valor “France” por “US” en la petición XQuery, y obtener una secuencia de 8 artículos.

6.2.4 COPIAR, RENOMBRAR, Y BORRAR COLECCIONES Y DOCUMENTOS

En Qizx Studio, estas tareas son realmente fáciles de realizar: sólo haciendo clic con el botón derecho sobre el miembro de la biblioteca a copiar, renombrar o borrar y selecciona la propiedad del menú del artículo.

Para copiar y renombrar, tenemos que indicar un path destino: este path debería estar dentro de una colección existente y no debería apuntar a un objeto existente.

6.3 PROGRAMANDO CON QIZX API

Al igual que se ha ido haciendo con Qizx Studio, se verá como crear una biblioteca y como se van introduciendo elementos dentro de ella.

6.3.1 CREAR UNA BIBLIOTECA (LIBRARY)

Para crear una biblioteca (Library), se debe primero crear un gestor de bibliotecas (LibraryManager) y con este se puede crear una biblioteca (Library). Para ver como se puede realizar esto, se van a mostrar dos métodos que se encargan de la creación de un gestor de bibliotecas (LibraryManager) y una biblioteca (Library) respectivamente:

```
private static LibraryManager getLibraryManager(File storageDir)

throws IOException, QizxException

{

LibraryManagerFactory factory =
LibraryManagerFactory.getInstance();

if (storageDir.exists())

{

      return factory.openLibraryGroup(storageDir);

}

Else

{

      if (!storageDir.mkdirs())

      {

            throw new IOException("cannot create directory '" +

            storageDir + "'");

       }

            System.out.println("Creating  library  group  in  '" +
      storageDir + "'...");

       return factory.createLibraryGroup(storageDir);

}

}
```

A Library Manager almacena todos sus datos en un único directorio del sistema de ficheros. Creando “LibraryManager” automáticamente crea este directorio si no existe.

A LibraryManager es obtenido utilizando los métodos “openLibraryGroup” o “createLibraryGroup” de un LibraryManagerFactory. El LibraryManagerFactory es obtenido directamente de “LibraryManagerFactory.getInstance”.

Una vez que se tiene el LibraryManager, se puede crear la Library (Biblioteca), con el siguiente método:

```
private static Library getLibrary(LibraryManager libManager, String libName)

throws QizxException

{

Library lib = libManager.openLibrary(libName, /*user*/ null);

if (lib == null)

{

      System.out.println("Creating library '" + libName +"'...");

      lib = libManager.createLibrary(libName, /*user*/ null);

}

return lib;

}
```

1 openLibrary especificando el nombre de la biblioteca (library) devuelve dicha biblioteca. Devolverá null si dicha biblioteca no existe.

2 createLibrary crea y devuelve la biblioteca que le especificamos con su nombre.

6.3.2 CREAR UNA COLECTION E IMPORTAR DOCUMENTOS EN ELLA

Una vez que se tiene creada la biblioteca (Library) se puede importar documentos dentro de ella, pero también se puede crear colecciones e importar los documentos sobre estas colecciones. Se verá como crear una colección y luego se importará los documentos de un directorio dentro de esta colección. Se puede ver en el siguiente fragmento de código:

```
System.out.println("Obtenemos una collection...");

Collection collection = lib.getCollection(path_collection);

if(collection == null)

{
```

```
            System.out.println("Estamos dentro collection == null");

            collection = lib.createCollection(path_collection);

            if(collection == null)

                 System.out.println("Collection es null");

                                //

             else  //la collection se ha creado con Éxit

             {

                   System.out.println("Nuestra    Collection:    "    +
             collection.getName());

             System.out.println("Nuestra collection está en el path: "
             + collection.getPath());

             }

      }
```

En el fragmento anterior, se ha creado una colección en el caso de que dicha colección no existiera y luego se ha obtenido su nombre y su path. A continuación se va a importar los ficheros de una lista a la colección que antes se había creado:

```
      System.out.println("Vamos a importar todos los ficheros a la
base de datos...");

      i=0;

      String path_fichero = "";

      while(lista_ficheros_xml[i]!=null)

      {

      System.out.println("Importamos      el      fichero:      "      +
      lista_ficheros_xml[i].getName());

            path_fichero     =     path_collection     +     "/"     +
      lista_ficheros_xml[i].getName();

            System.out.println("Path    fichero    collection:    "    +
path_fichero);

            lib.importDocument(path_fichero,lista_ficheros_xml[i]);

            i++;

      }

            lib.commit();

            lib.close();

            libManager.closeAllLibraries(1000);
```

Para importar un fichero dentro de la biblioteca se usa el método importDocument de la clase Library, especificando el path del fichero y su nombre.

Una vez importados los documentos, se tiene que ejecutar el método commit(), para que los cambios sean efectivos dentro de la base de datos. Por último, se cerrará la base de datos con close() (se cierra la biblioteca) y closeAllLibraries() (se cierra toda la base de datos).

De esta manera, se tiene una biblioteca (base de datos) llena de ficheros. El siguiente usuario que quiera realizar búsquedas dentro de estos documentos, realizará peticiones XQuery, a los documentos que fueron importados con anterioridad.

6.3.3 PETICIONES XQUERY SOBRE LA BIBLIOTECA

Para hacer peticiones sobre la base de datos, los se puede hacer fácilmente utilizando los siguientes métodos:

```
Expression expr = lib.compileExpression(script);

ItemSequence results = expr.evaluate();

while (results.moveToNextItem()) {

Item result = results.getCurrentItem();

/*Do something with result.*/

}
```

Primero se compila la expresión usando “Library.compileExpression”. Si no se tiene errores en la compilación (CompilationException), devolverá un objeto de tipo Expression.

Luego se evalúa la expresión usando “Expression.evaluate”. Si no existe errores (EvaluationException), se obtendrá como resultado de la evaluación un “ItemSequence”. A partir de aquí utilizando los métodos “moveToNextItem()” y “getCurrentItem()” se podrá obtener los “Item”s de la búsqueda.

El script que se le pasa a compileExpression es un fichero .xql que se ha pasado a un tipo String. En el fichero .xql se escribirá la expresión que se quiera, para que se ejecute en la base de datos.

Más detalles de expresiones .xql se puede ver en el capítulo de XQuery.

Paso de parámetros

Con XQuery existe la posibilidad de pasar parámetros antes de la evaluación. Para ello, se utilizan ficheros .xql que contienen variables externas. El valor de dichas variables se pasarían como parámetros. Además, se puede indicar a posteriori la colección que se quiera donde vaya a hacerse la petición o búsqueda XQuery en lugar de especificarlo en el fichero .xql. Un ejemplo de todo esto se puede ver a continuación:

```
      private    static    Expression    compileExpression(Library
lib,String script,LibraryMember queryRoot,QName[] varNames,String[]
varValues)

throws IOException, QizxException

{

           Expression expr;

     try

     {

         expr = lib.compileExpression(script);

           }

           catch (CompilationException e)

     {

         Message[] messages = e.getMessages();

         For (int i = 0; i < messages.length; ++i) {

         error(messages[i].toString());

     }

     throw e;

      }

      if (queryRoot != null)

     expr.bindImplicitCollection(queryRoot);

      if (varNames != null)

      {

     for (int i = 0; i < varNames.length; ++i)

     {

     expr.bindVariable(varNames[i], varValues[i], /*type*/ null);

     }

}

return expr;

   }
```

1 “Expression.bindImplicitCollection” permite realizar peticiones conteniendo paths que no están prefijados con Collection(“XXX”) ó doc(“YYY). De esta forma, si

se utiliza la expresión de bindImplicitCollection en el código, se puede escribir el siguiente fichero .xql:

```
(: List all books by their titles. :)

declare namespace t = "http://www.qizx.com/namespace/Tutorial";

//t:book/t:title
```

En lugar del siguiente:

```
(: List all books by their titles. :)

declare namespace t = "http://www.qizx.com/namespace/Tutorial";

collection("/Books")//t:book/t:title
```

En este último fragmento se ha tenido que especificar la colección donde se quiera que se haga la búsqueda, ya que no se especificó antes.

2 "Expression.bindVariable" permitirá dar valores a variables que en el fichero .xql se ha definido como variables externas (parámetros). Un ejemplo de fichero .xql donde se utiliza las variables externas, cuyos valores se pasan luego como parámetros, usando el método bindVariable, se puede ver a continuación:

```
  (: List all books containing the value of variable $searched

in their titles. :)

declare namespace t = "http://www.qizx.com/namespace/Tutorial";

declare variable $searched external;

collection("/Books")//t:book/t:title[contains(., $searched)]
```

6.3.4 BORRAR DOCUMENTOS Y COLECCIONES DE UNA BIBLIOTECA

Para borrar uno o más documentos o colecciones dentro de una base de datos, se utilizará el método deleteLibrary de la clase LibraryManager. El path del documento o la colección es especificado. Si no se especifica ningún path, toda la biblioteca (Library) será borrada. Un ejemplo para borrar una biblioteca sería:

```
libManager.deleteLibrary(libName);
```