

## 4.6. Trabajar con la librería de BaseX para Java



# Índice

---

Objetivos .....	3
BaseX XQJ API .....	4
Introducción a BaseX XQJ API .....	4
Obtener BaseX XQJ API .....	5
Ejecutar consultas XQuery desde Java .....	6
Obtener objetos Node .....	11
Sentencias de actualización XQuery .....	12
Despedida .....	14
Resumen .....	14

# Objetivos

En esta lección perseguimos los siguientes objetivos:

- Desarrollar aplicaciones Java que accedan a bases de datos XML a través del API BaseX XQJ que la aplicación BaseX pone a nuestra disposición.
- Ejecutar consultas XQuery desde programas Java.

# BaseX XQJ API

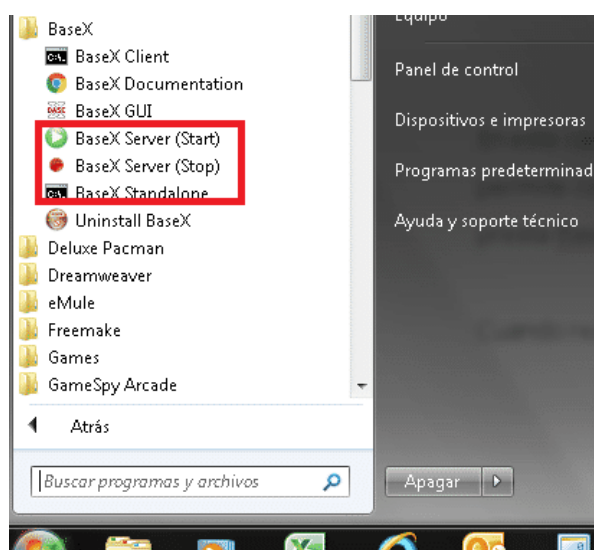
## Introducción a BaseX XQJ API

Existen varias librerías o APIs en el mercado cuyo objeto es facilitar la explotación de documentos XML desde programas Java.

Su función principal es facilitar la **ejecución de consultas XQuery** y la obtención de los resultados para su posterior tratamiento.

En este caso, trabajaremos con la librería que nos brinda la aplicación BaseX (BaseX XQJ API) y que permite comunicar nuestro programa Java con el servicio de acceso a bases de datos XML, que presta BaseX en su modelo de arquitectura Cliente/Servidor.

Cuando realizaste la instalación de BaseX en tu equipo, además del ejecutable *BaseX GUI*, se instalaron otros programas, tal y como puedes ver en la imagen:



Otros programas instalados durante la instalación de BaseX.

Entre los programas instalados, hay que resaltar dos que tienen que ver especialmente con la arquitectura Cliente/Servidor de BaseX:

- **BaseX Server (Start):** inicia sesión con el servidor de BaseX, que quedará a la escucha de las peticiones de los clientes que requieren acceso a las bases de datos a través de un *host* y puerto.
- **BaseX Server (Stop):** cierra sesión con el servidor de BaseX.

Nuestra aplicación Java, haciendo uso de la librería BaseX XQJ API, actuará como cliente, realizando solicitudes al servidor de BaseX a través del *host* y el puerto.

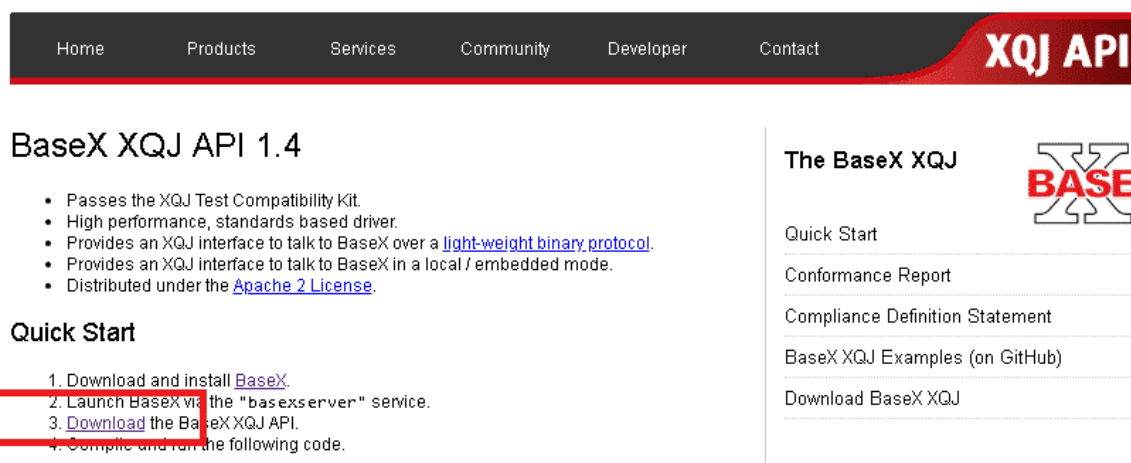
## Obtener BaseX XQJ API

Sigue los pasos que te vamos a exponer en este apartado para **obtener BaseX XQJ API**.

### 1. Accede a BaseX XQJ API

Pulsa el botón de la derecha para acceder a la web. <http://xqj.net/basex/>

### 2. Haz clic en el enlace *Download*.



BaseX XQJ API 1.4

- Passes the XQJ Test Compatibility Kit.
- High performance, standards based driver.
- Provides an XQJ interface to talk to BaseX over a [light-weight binary protocol](#).
- Provides an XQJ interface to talk to BaseX in a local / embedded mode.
- Distributed under the [Apache 2 License](#).

**Quick Start**

1. Download and install [BaseX](#).
2. Launch BaseX via the "basexserver" service.
3. [Download](#) the BaseX XQJ API.
4. Compile and run the following code.

**The BaseX XQJ**

[Quick Start](#)

[Conformance Report](#)

[Compliance Definition Statement](#)

[BaseX XQJ Examples \(on GitHub\)](#)

[Download BaseX XQJ](#)

### 3. Dentro de la lista de descargas, haz clic en la primera opción, que actualmente es la versión 1.4.0.

#### BaseX XQJ via ZIP Packages

If you're not using Maven, you can download a zip package which contains all necessary jars to use BaseX XQJ. If you are planning on using BaseX XQJ in embedded mode, BaseX jars will also need to be included on the classpath at runtime.

Version	Build Date	Download	Size
1.4.0	11th February 2015	<a href="#">basex-xqj-1.4.0.zip</a>	332 kb
1.3.0	10th February 2014	<a href="#">basex-xqj-1.3.0.zip</a>	323 kb
1.2.3	1st October 2013	<a href="#">basex-xqj-1.2.3.zip</a>	682 kb
1.2.2	4th September 2012	<a href="#">basex-xqj-1.2.2.zip</a>	682 kb
1.2.1	23rd August 2012	<a href="#">basex-xqj-1.2.1.zip</a>	682 kb
1.2.0	8th June 2012	<a href="#">basex-xqj-1.2.0.zip</a>	677 kb

4. Espera a que se complete la descarga y guarda el archivo obtenido en la ubicación que desees. Si no sabes dónde está el archivo, seguro que lo tendrás en la carpeta *descargas* de tu equipo. El archivo obtenido se llamará *basex-xqj-1.4.0.zip* o algo similar.

5. Descomprime el archivo para obtener una carpeta con nombre *basex-xqj-1.4.0*, donde estará ubicado el API.

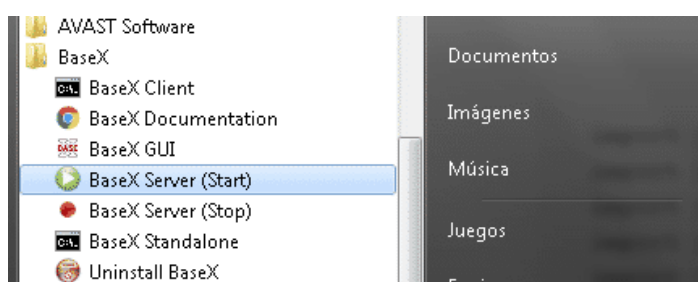
## Ejecutar consultas XQuery desde Java

BaseX puede activarse como un **servicio para acceder de forma remota a las bases de datos XML** que administra.

Utilizaremos este servicio para obtener información sobre la base de datos ALMACEN, que creaste en la lección anterior.

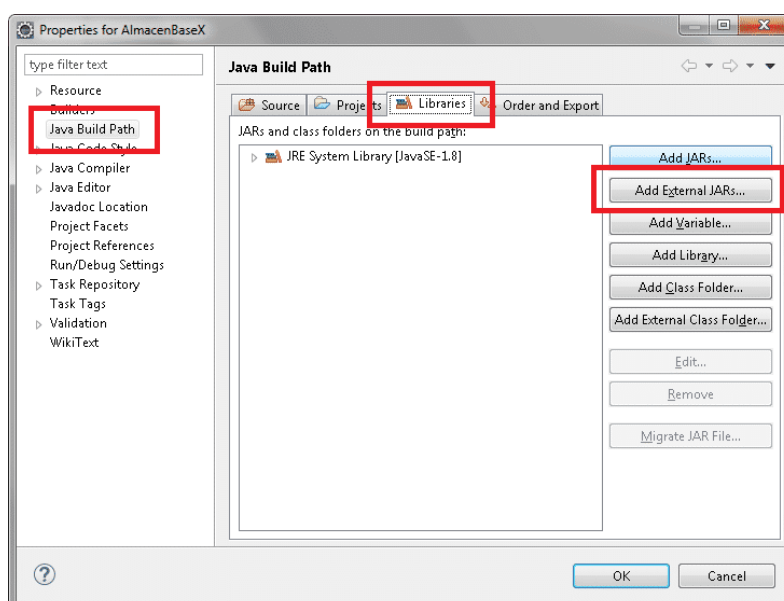
¡Ojo! Antes que nada, tendrás que dejar iniciado el servidor de BaseX que presta el servicio.

Accede a la carpeta de programas *BaseX* y ejecuta *BaseX Server (Start)*.



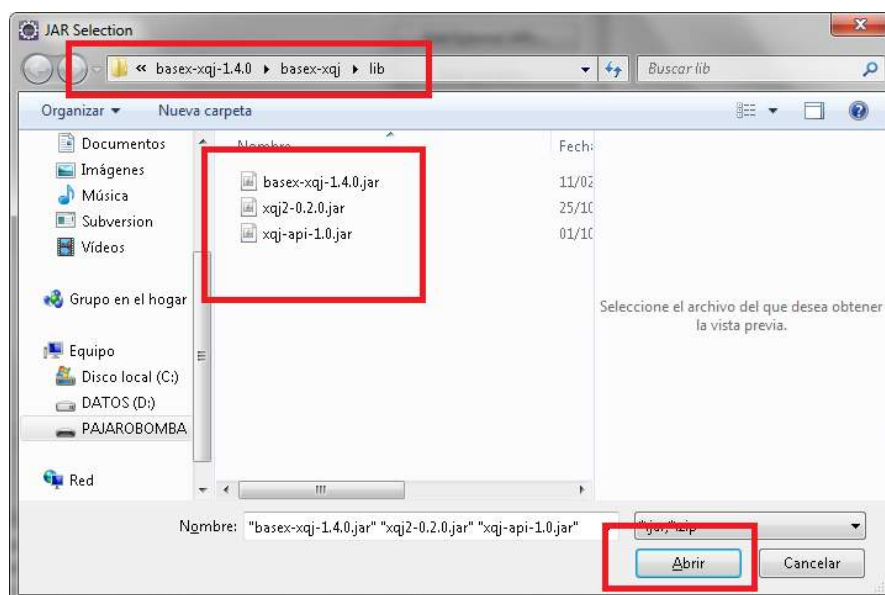
Una vez iniciado el servicio, sigue los pasos que te indicamos a continuación:

1. Crea un proyecto Java llamado *AlmacenBaseX* en Eclipse.
2. Dentro del proyecto, importa la librería *BaseX XQJ API* que descargaste en el apartado anterior. Para importar la librería, haz clic derecho sobre el nombre del proyecto y selecciona en el menú contextual la opción *Properties*. Dentro de las propiedades, tendrás que situarte en *Java Build Path* y en la pestaña *Libraries*.

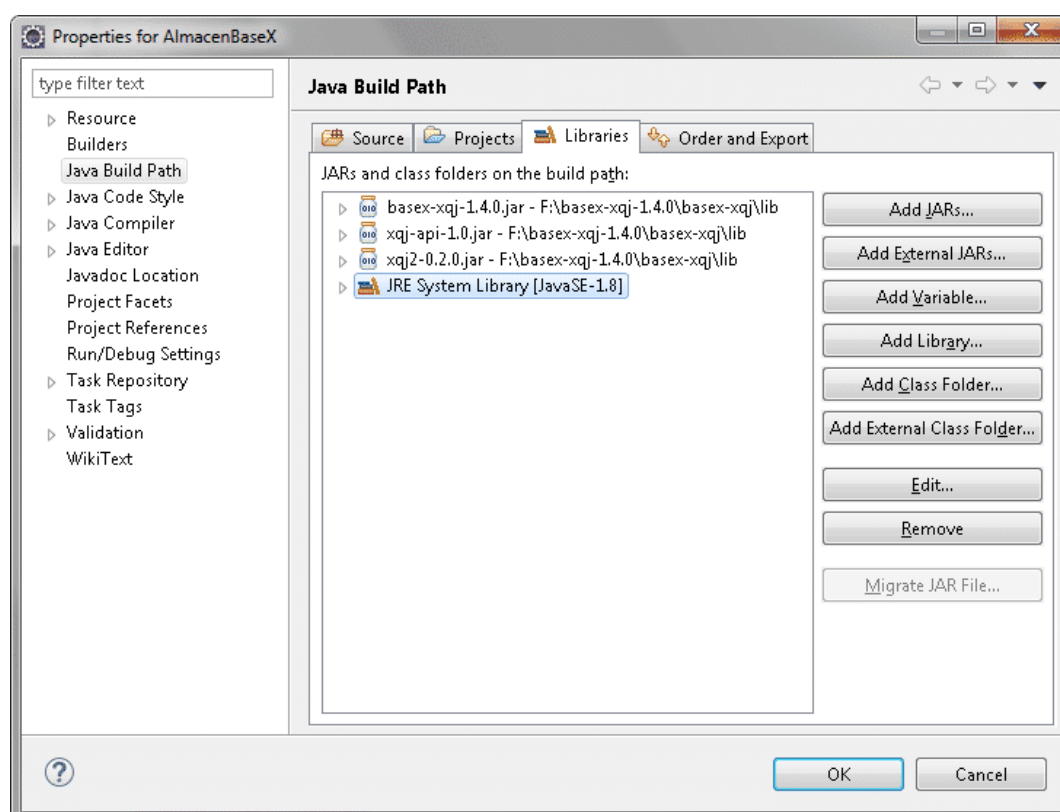




Haz clic en el botón *Add External JARs...* En el cuadro de diálogo *JAR Selection*, sitúate en la ubicación donde descargaste la librería y busca la carpeta *lib*. La ubicación será algo así: *basex-xqj-1.4.0\basex-xqj\lib*. Después, selecciona los tres archivos *.jar* que aparecerán y haz clic en el botón *Abrir*.



Finalmente, el cuadro de diálogo *Properties* quedará así:



**3.** Crea la clase *Principal* con el código que aparece a continuación; después, la analizaremos detenidamente para que comprendas cómo funciona la librería *BaseX XQJ API*.

```
import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQExpression;
import javax.xml.xquery.XQResultSequence;
import net.xqj.basex.BaseXXQDataSource;

public class Principal {
    public static void main(String[] args) {
        XQDataSource xds = new BaseXXQDataSource();
        XQConnection con;
        XQExpression expr;
        XQResultSequence result;
        String sentencia;

        try {
            xds.setProperty("serverName", "localhost");
            xds.setProperty("port", "1984");
            con = xds.getConnection("admin", "admin");
        } catch (XQException e) {
            System.out.println("Error al establecer la conexión con
BaseX");
            System.out.println(e.getMessage());
            return;
        }
        System.out.println("Establecida la conexión con BaseX");
        sentencia = "for $pro in fn:collection('almacen')//productos return
$pro/producto";

        try {
            expr = con.createExpression();
            result = expr.executeQuery(sentencia);
        } catch (XQException e) {
            System.out.println("Error al ejecutar la sentencia XQuery");
            System.out.println(e.getMessage());
            return;
        }

        try {
            while (result.next()) {
                System.out.println(result.getItemAsString(null));
            }
        } catch (XQException e) {
            System.out.println("Error al recorrer los elementos
obtenidos");
            System.out.println(e.getMessage());
        }

        try {
            con.close();
        } catch (XQException e) {
            System.out.println("Error al cerrar la conexión con BaseX");
            System.out.println(e.getMessage());
        }
    }
}
```



En primer lugar, observa que estamos utilizando nuevas clases, ubicadas en dos paquetes diferentes:

## **javax.xml.xquery**

Procesador XQuery para Java que contiene los recursos necesarios para acceder a la base de datos XML requerida y ejecutar sentencias XQuery.

## **net.xqj.basex**

A este paquete pertenece la clase `BaseXXQDataSource` (origen de datos ligado a BaseX) que implementa la interfaz genérica `XQDataSource`.

**Ahora, vamos a estudiar el código paso a paso:**

### **Establecimiento de la conexión con BaseX:**

```
XQDataSource xds = new BaseXXQDataSource();
XQConnection con;
XQExpression expr;
XQResultSequence result;
String sentencia;

try {
    xds.setProperty("serverName", "localhost");
    xds.setProperty("port", "1984");
    con = xds.getConnection("admin", "admin");
} catch (XQException e) {
    System.out.println("Error al establecer la conexión con BaseX");
    System.out.println(e.getMessage());
    return;
}
System.out.println("Establecida la conexión con BaseX");
```

Lo primero que hacemos es declarar una referencia a un objeto de tipo ***XQDataSource***, pero ***XQDataSource*** no es una clase, sino una interfaz genérica que es implementada por varias clases, cada una de las cuales representa a un determinado gestor de bases de datos XML. ***BaseXXQDataSource*** es una clase directamente relacionada con BaseX; para otra aplicación se utilizaría otra clase diferente.

Con los métodos `setProperty()` estamos designando los valores para establecer la conexión con XBase. Estos valores son: ***serverName***, que en nuestro caso es ***localhost*** y ***port*** o puerto por el que escucha BaseX, que es ***1984***.

Por último, establecemos la conexión con el método ***getConnection()*** que recibe dos argumentos: ***login*** y ***password*** cuyos valores para BaseX son, por defecto, ***admin*** y ***admin***.

```

sentencia = "for $pro in fn:collection('almacen')//productos return
$pro/producto";

try {
    expr = con.createExpression();
    result = expr.executeQuery(sentencia);
} catch (XQException e) {
    System.out.println("Error al ejecutar la sentencia XQuery");
    System.out.println(e.getMessage());
    return;
}

```

La clase ***XQExpresion*** nos permite ejecutar sentencias XQuery por medio de su método ***executeQuery()***, que recibe como argumento una cadena con la sentencia a ejecutar y devuelve un objeto ***XQResultSequence***, que nos permite acceso secuencial a los elementos resultado.

La expresión *XPath* de acceso a los elementos XML contiene la expresión ***fn:collection('almacen')***. Con dicha expresión indicamos el nombre de la base de datos a la que deseamos acceder, puesto que sabemos que BaseX nos da acceso a distintas bases de datos.

```

try {
    while (result.next()) {
        System.out.println(result.getItemAsString(null));
    }
} catch (XQException e) {
    System.out.println("Error al recorrer los elementos obtenidos");
    System.out.println(e.getMessage());
}

```

Nuestro objeto *result*, de tipo ***XQResultSequence*** dispone del método ***next()*** que nos permite ir avanzando al siguiente elemento o nodo dentro de la estructura XML resultado de la ejecución de la sentencia XQuery. Con el método ***getItemAsString()*** obtenemos todo el contenido XML de cada elemento; ***getItemAsString()*** recibe como argumento parámetros de serialización, pero en la mayoría de los casos es suficiente con pasar el valor *null*.

```

try {
    con.close();
} catch (XQException e) {
    System.out.println("Error al cerrar la conexión con BaseX");
    System.out.println(e.getMessage());
}

```

Por último, debemos cerrar la conexión con BaseX.

Ahora puedes probar a modificar el contenido de la variable *sentencia* para poner a prueba otros ejemplos XQuery, tal y como aprendiste en la lección anterior.

## Obtener objetos Node

Vamos a modificar el programa anterior para obtener, por cada producto iterado, un objeto *Node* que representará el árbol DOM de cada elemento.

Recuerda que la clase *Node* pertenece al *parser* DOM que estudiaste en la segunda lección de esta unidad.

```
import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQExpression;
import javax.xml.xquery.XQResultSequence;

import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import net.xqj.basex.BaseXXQDataSource;

public class Principal {
    public static void main(String[] args) {
        XQDataSource xds = new BaseXXQDataSource();
        XQConnection con;
        XQExpression expr;
        XQResultSequence result;
        String sentencia;

        try {
            xds.setProperty("serverName", "localhost");
            xds.setProperty("port", "1984");
            con = xds.getConnection("admin", "admin");
        } catch (XQException e) {
            System.out.println("Error al establecer la conexión con BaseX");
            System.out.println(e.getMessage());
            return;
        }

        System.out.println("Establecida la conexión con BaseX");
        sentencia = "for $pro in fn:collection('almacen')//productos return $pro/producto";

        try {
            expr = con.createExpression();
            result = expr.executeQuery(sentencia);
        } catch (XQException e) {
            System.out.println("Error al ejecutar la sentencia XQuery");
            System.out.println(e.getMessage());
            return;
        }

        int contador = 0;
        try {
            while (result.next()) {
                contador++;
                System.out.println("Producto nº " + contador);
                Node nodoProducto = result.getNode();
                mostrarProducto(nodoProducto);
            }
        } catch (XQException e) {
            System.out.println("Error al recorrer los elementos obtenidos");
            System.out.println(e.getMessage());
        }

        try {
            con.close();
        } catch (XQException e) {
            System.out.println("Error al cerrar la conexión con BaseX");
        }
    }
}
```

```

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

    private static void mostrarProducto(Node nodo) {
        NodeList nodos = nodo.getChildNodes();
        for (int i=0; i<nodos.getLength();i++) {
            Node nodoHijo = nodos.item(i);
            if (nodoHijo.getNodeType() == Node.ELEMENT_NODE) {
                System.out.println(nodoHijo.getNodeName() + ": " +
nodoHijo.getTextContent());
            }
        }
    }
}

```

## Sentencias de actualización XQuery

XQuery da soporte completo también para tareas de actualización de documentos XML, permitiendo no sólo la obtención de nodos, sino también su **borrado, actualización o inserción**.

Puedes realizar unas pruebas desde el editor XQuery de BaseX.

```

insert node
<categoria id="9">
  <nombreCategoria>Legumbres</nombreCategoria>
  <productos>
    <producto>
      <nombreProducto>Garbanzos del Bierzo</nombreProducto>
      <cantidadPorUnidad>Paquete de un kg</cantidadPorUnidad>
      <precio>20.16</precio>
      <stock>39</stock>
    </producto>
  </productos>
</categoria>
after /almacen/categoria[8]

```

Esta sentencia añade la nueva categoría *legumbres* con su producto "Garbanzos del Bierzo" y lo sitúa justo después de la octava categoría.

```

insert node
  <producto>
    <nombreProducto>Judiones del Bierzo</nombreProducto>
    <cantidadPorUnidad>Paquete de un kg</cantidadPorUnidad>
    <precio>18.16</precio>
    <stock>12</stock>
  </producto>
after /almacen/categoria[last()]/productos/producto[last()]

```

Esta sentencia añade el nuevo producto "Judiones del Bierzo" detrás del último producto de la última categoría que es *Legumbres*.

```
replace value of node  
/almacen/categoria[last()]/productos/producto[last()]/precio  
with 3
```

Esta sentencia asigna el valor 3 al precio del último producto de la última categoría.

```
delete node /almacen/categoria[last()]
```

Esta sentencia elimina la última categoría, que para nuestro documento es *Legumbres*.

# Despedida

## Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

- **BaseX XQJ API** es una librería suministrada por BaseX que actúa como intermediaria entre un programa Java y las bases de datos que suministra el servidor de BaseX.
- La principal función de la librería BaseX XQJ API es permitir **ejecutar sentencias *XQuery* desde aplicaciones Java**.
- Podemos obtener la representación DOM de los elementos obtenidos a través de la ejecución de una sentencia *XQuery* como objeto de la clase *Node*.