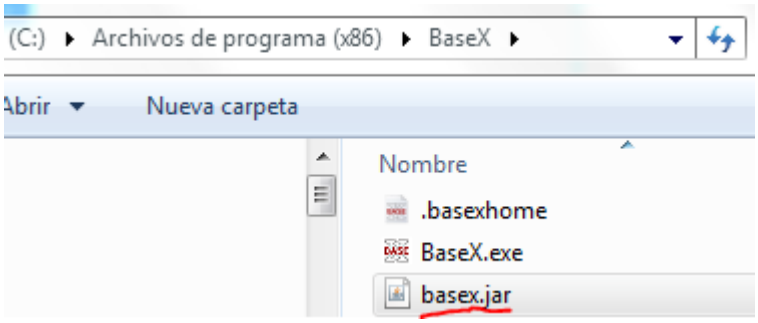
Tema 6. Anexo 2 – XPath y XQuery desde Java

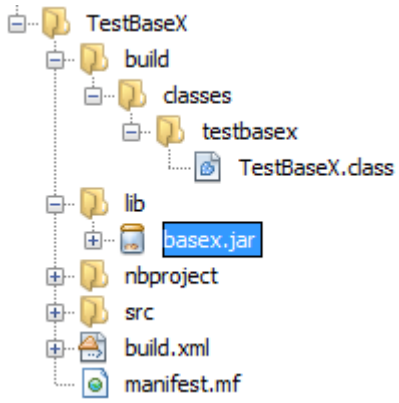
6A2.1. Librerías

Vamos a ver cómo hacer uso del sistema gestor BaseX desde Java, particularmente en el entorno NetBeans. El primer paso consiste en habilitar un proyecto nuevo e incorporar en él las librerías oportunas.

El paquete con el que vamos a poder realizar las operaciones fundamentales de acceso a las bases de datos es "basex.jar" y lo encontramos en la misma raíz de la carpeta de instalación del programa. Atención: no está dentro de la carpeta "lib", aunque pudiera pensarse que así es.



Debemos crear un proyecto nuevo en NetBeans e incorporar este paquete a las librerías del proyecto. Para este caso sí que podemos habilitar una carpeta "lib" y meterlo ahí:



Para nuestro guión, hemos preparado un proyecto llamado "TestBaseX". El cual se basa en el asistente para aplicación Java (incorpora método estático "main").

En la ilustración de la derecha puede verse la vista "Archivos" de este proyecto. La clase principal cuelga del paquete por defecto "testbasex". No hay más archivos fuente.

La librería, como se ha dicho, queda ubicada dentro de la carpeta "lib", que se ha habilitado a tal efecto.

El siguiente paso es realizar los "imports" necesarios. A continuación se muestran los que vamos a necesitar para los primeros ejemplos:

```
import org.basex.core.*;
import org.basex.core.cmd.*;
import org.basex.util.options.Options;
```

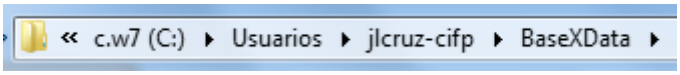
6A2.2. Contexto y conexión

El acceso a bases de datos suele requerir un objeto referido al "contexto" de la conexión. Ocurre con muchos sistemas gestores y librerías de conectividad. También con BaseX.

Una particularidad de BaseX es que la ruta en la que se almacenan las bases de datos no se establece en el momento de crear el contexto, sino que es una opción más global. Además, por defecto, esta ruta no está establecida en la ubicación predeterminada de las bases de datos (al final del punto 6.2.3 vimos que la ubicación predeterminada era la subcarpeta "data" de la ruta en la que se instaló BaseX, que por defecto suele ser "C:\Program Files\BaseX" o "C:\Program Files (x86)\BaseX" en sistemas de 64 bits).

Recordemos que cuando hablamos de esta ruta NO estamos hablando del almacén de ficheros XML, sino de los archivos auxiliares que sirven como índices y metadatos sobre las bases de datos. En nuestro ejemplo, los archivos XML siguen colgando de "C:\xmldb\...".

A todo esto, las librerías de Java esperan que la ruta raíz para las bases de datos XML se sitúe en la carpeta del perfil de usuario, en la subcarpeta "BaseXData". En esta captura puede verse un ejemplo:



De hecho, si no especificamos otra cosa y creamos una base de datos desde una aplicación Java, entonces la base de datos se ubicará ahí.

Para nuestro ejemplo, por seguir con "xmlregantes", vamos a cambiar este comportamiento. En una situación real, las bases de datos deberían colgar de una carpeta que no estuviera dentro del perfil del usuario ni tampoco dentro de archivos de programa.

Manos a la obra, las primeras instrucciones de nuestro programa Java podrían ser estas:

```
public static void main(String[] args) throws BaseXException
{

    // Especificar la ubicación raiz para los archivos de apoyo a bases de
    // datos XML, coincidiendo con la predeterminada por el visor de BaseX
    Options.setSystem(StaticOptions.DBPATH.name(),
            "C:\\Program Files (x86)\\BaseX\\data");
    // Creación del contexto de conexión a la base de datos
    Context context = new Context();
```

La primera instrucción cambia una variable de entorno usada por el framework de acceso a BaseX, llamada "DBPATH", que es la que indica la ruta base que hemos mencionado.

La segunda instrucción crea el objeto "context" como de la clase "Context". Haremos referencia a este objeto cada vez que tengamos que hacer una operación en la base de datos.

Es importante realizar estas operaciones en este orden, ya que de lo contrario el contexto habrá tomado la ruta predeterminada para las bases de datos y, aunque se modifique después, ya será tarde para él.

La siguiente tarea a realizar es aperturar una base de datos en concreto, de entre las que puedan estar registradas en la ruta apuntada por DBPATH. Se consigue así:

```
// Abrir la base de datos concreta que queramos usar:
new Open("xmlregantes").execute(context);
```

Como ya adelantamos, ha sido necesaria una referencia al objeto "contexto".

A partir de este momento ya es posible realizar operaciones de consulta a la base de datos XML de regantes, cuyos ficheros recordemos que están en "C:\xmldb\xmlregantes\".

6A2.3. Ejecución de consultas

Parta realizar consultas e iterar un proceso sobre sus resultados, necesitamos como poco estos "imports":

```
import org.basex.query.*;
import org.basex.query.iter.*;
import org.basex.query.value.item.*;
```

A continuación se muestra un ejemplo completo de cómo plantear una consulta XQuery y de cómo acceder al conjunto de valores devueltos por la misma. La consulta recorrerá los nombres de todos los regantes, mediante una expresión FLWOR.

```
44          // Consulta que recorrerá los nombres de los regantes,
45          // para mostrarlos por la salida estándar
46          String q = "for $x in db:open('xmlregantes')"
47                  +"//datos//regantes//nombre return data($x)";
48          QueryProcessor qp = new QueryProcessor (q,context);
49          try
50          {
51              Iter ip = qp.iter();
52              for (Item item; (item = ip.next())!=null;) {
53                  System.out.println(item.toJava());
54              }
55          }
56          catch (QueryException ex)
57          {
58              System.out.println("ERROR: "+ex.getMessage());
59          }
```

```
Salida
TestBaseX (debug)  Consola del depurador
debug:
Mariano Iglesias
Sebastian Alonso
Amancio Gasset
BUILD SUCCESSFUL (total time: 1 second)
```

La captura de excepciones de tipo “QueryException” es un requisito de algunos métodos usados. La base de las consultas se apoya en dos clases fundamentales:

- **QueryProcessor**: que es la que permite indicar y ejecutar la consulta, proporcionando el acceso a un cursor con el que recabar los datos devueltos.
- **Iter**: que proporciona los mecanismos para recuperar los datos a través del cursor proporcionado por el objeto procesador de consultas, de la clase QueryProcessor.

6A2.4. Otras operaciones

En la siguiente captura pueden verse otras operaciones relativas a la creación y registro de bases de datos, creación o regeneración de índices, mostrar información sobre la base de datos y la clausura de la conexión con la base de datos abierta:

```
// Crear una base de datos a partir de un conjunto de archivos XML
new CreateDB("xmlregantes2","C:\\xmldb\\xmlreg2\\").execute(context);

// Crear un índice de texto completo sobre la base de datos
new CreateIndex("fulltext").execute(context);

// Mostrar información sobre la base de datos
System.out.print(new InfoDB().execute(context));

// Cerrar la conexión con la base de datos abierta
new Close().execute(context);
```