BaseX XQJ Documentation

Version: 1.1.0, 5th June 2012

http://xqj.net/basex

Introduction to BaseX XQJ

BaseX XQJ is an interface which can communicate with a BaseX XML Database. Importantly BaseX XQJ is a library which implements the XQuery API for Java™ [1] interfaces outlined by the Java Community Process (JCP).

The XQuery API for Java™ Specification package name is `javax.xml.xquery`

Overview of BaseX XQJ

BaseX XQJ is used to communicate with a BaseX XML Database from a Java environment.

The XQuery API for Java™ (XQJ) defines a standard way of connecting to an XML Database/DataSource, submitting XQuery expressions and processing their results within the Java environment. In essence, the XQJ API specification is very similar in its approach to the highly successful Java Database Connectivity API (JDBC). The JDBC API specification defines how a Java client should connect to a Relational Database, submit SQL queries and process their results within the Java environment. BaseX XQJ is designed primarily as an "industry standard interface" to the BaseX XML Database from the Java environment.

BaseX XQJ communicates with a BaseX XML Database via a TCP/IP Socket

BaseX XQJ requires the BaseX XML Database to listen for incoming connections on a port, e.g. port 1984.

An BaseX XML Database provides a low-level network based interface, allowing external programming environments (outside of BaseX) to communicate with a BaseX XML Database.

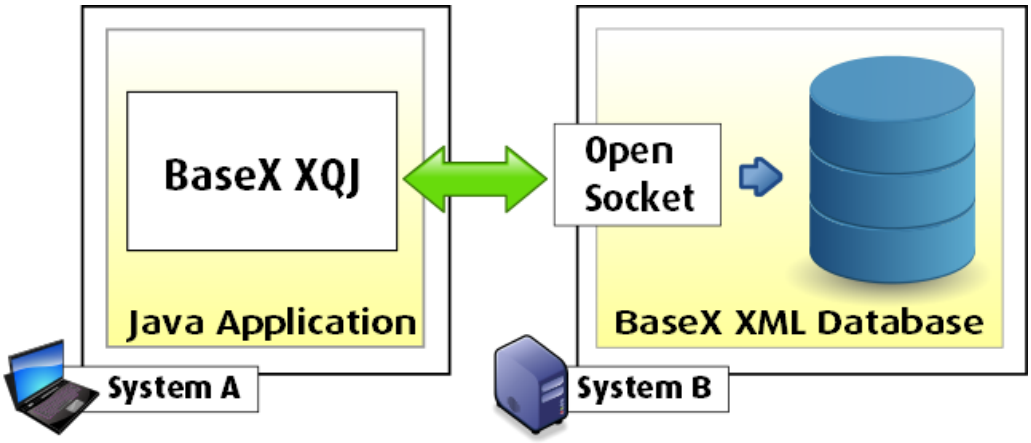
BaseX XQJ communicates with BaseX by submitting XQuery statements over the wire and processing the results.

BaseX XQJ enabled applications can be written either as standalone or as part of a application server environment.

Client-Server Architecture

BaseX XQJ communicates with a BaseX Server via a client-server architecture, where BaseX XQJ acts as the client and the BaseX XML Database acts as the server.

The following diagram illustrates this concept at a high level:



BaseX XQJ enabled application can reside on the same host as the server, or providing systems are networked, they can reside on separate hosts.

In the diagram, BaseX XQJ is used as part of a Java Application on System A.

The application submits XQuery statements to a on System B by connecting to a BaseX Server configured on the BaseX XML Database.

The BaseX Server evaluates these queries against a BaseX Database which the server instance manages and returns results back to the client for processing.

1 XQuery API for Java™ at the JCP - http://jcp.org/en/jsr/detail?id=225

Connections

An XQConnection represents a physical, persistent connection between the client and the server.

It is **essential** that the `close()` method is called on `XQConnection` objects once they have been finished with.

BaseX XQJ (Server Requirements)

BaseX XQJ was built for BaseX version 7.2. Previous versions of BaseX XQJ may work but have not been tested.

BaseX XQJ (Client Requirements)

- The BaseX XQJ Driver requires Java 1.5 or later

BaseX XQJ has been tested against the Sun Java Virtual Machine. Other Java Virtual Machines may work but have not been tested.

BaseX XQJ binary jar file has no dependencies other than the XQJ and XQJ2 interfaces. Both of these dependencies are included in the package which contained the BaseX XQJ binary.

Programming with BaseX XQJ

Multi-tier applications which communicate with the BaseX XML Database as the content repository can be developed with BaseX XQJ. This chapter describes some core XQJ concepts and BaseX XQJ specific concepts.

XQJ Connections

- XQConnection objects can be generated from an XQDataSource instance
- The BaseX XQJ XQDataSource implementation is `net.xqj.basex.BaseXXQDataSource`

To create an XQDataSource instance, write the following

```
XQDataSource ds = new BaseXXQDataSource();
```

An XQDataSource can be used repeatedly to generate new XQConnection objects. Before creating XQConnection objects, the XQDataSource must be given information about the BaseX XML Database.

A BaseX XQDataSource instance must be given at least a host and a port, the following example tells an XQDataSource instance that the Server's hostname is "localhost" and the Server's port is "1984". Additionally username and password information is also passed.

```
XQDataSource ds = new BaseXXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "1984");
ds.setProperty("user", "bx-user");
ds.setProperty("password", "bx-pass");
```

Creating Connections with Authentication

XQDataSource contains two methods which can be invoked to create a new XQConnection instance

`public XQConnection getConnection()`
: Creates a new XQConnection object, giving no authentication information to the server.

`public XQConnection getConnection(String user, String password)`
: Creates a new XQConnection object, sending the given user / password pair to the server.

The following example creates a XQDataSource, describes the BaseX Server and opens a XQConnection to the BaseX Server with a username of "bx-user" and a password of "bx-pass".

```
XQDataSource ds = new BaseXXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "1984");

XQConnection conn = ds.getConnection("bx-user","bx-pass");
```

Programming Basics

To use the BaseX XQJ, the following tasks are generally executed in order.

- Import the required libraries.
- Create an `XQDataSource` object via the `BaseXXQDataSource` implementation.
- Pass host and port information of a BaseX Server to the `XQDataSource` object.
- Create a `XQConnection` object from the `XQDataSource` instance.
- Create a `XQExpression` or `XQPreparedExpression` object from the `XQConnection` instance.
- Submit an XQuery expression to BaseX via the `XQExpression` or `XQPreparedExpression` object.
- Executing an XQuery expression returns an `XQResultSequence` object, loop through its items with the `next()` method.
- Once finished with a `XQConnection`, call the its `close()` method.

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

import net.xqj.basex.BaseXXQDataSource;


XQDataSource ds = new BaseXXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "1984");

XQConnection xqc = ds.getConnection("bx-user","bx-pass");

XQExpression xqe = xqc.createExpression();
XQResultSequence rs = xqe.executeQuery("for $x in 1,2,3,4 return $x");

while(rs.next())
{
     String stringValue = rs.getItemAsString(null);
     System.out.println(stringValue);
}
xqc.close();
```

XQJ Conformance

The XQJ Specification states that XQJ implementations should pass the Test Compatibility Kit (TCK) and provide a Compliance Definition statement outlining how all aspects of the XQJ implementation which are described by the XQJ Specification as "implementation-defined" have been implemented.

Technology Compatibility Kit (TCK) Conformance

The XQJ Specification provides a Technology Compatibility Kit (TCK).

The TCK is a piece of software which can be used to test whether or not an XQJ implementation complies with the XQJ specification.

Official XQJ TestCase	BaseX XQJ
SignatureTest	PASS (1/1)
XQCancelledExceptionTest	PASS (1/1)
XQConnectionTest	PASS (17/17)
XQDataFactoryTest	PASS (33/33)
XQDataSourceTest	PASS (11/11)
XQDynamicContextTest	FAIL (20/21)*
XQExceptionTest	PASS (5/5)
XQExpressionTest	PASS (9/9)
XQItemAccessorTest	PASS (20/20)
XQItemTest	PASS (2/2)
XQItemTypeTest	PASS (13/13)
XQMetaDataTest	PASS (25/25)
XQPreparedExpressionTest	PASS (9/9)
XQQueryExceptionTest	PASS (12/12)
XQResultItemTest	PASS (1/1)
XQResultSequenceTest	PASS (1/1)
XQSequenceTest	PASS (25/25)
XQSequenceTypeTest	PASS (3/3)
XQStackTraceElementTest	PASS (7/7)
XQStackTraceVariableTest	PASS (3/3)
XQStaticContextTest	PASS (35/35)
Pass Percentage	99.6%

Notes

* The XQJ 1.0 Specification states that the executing new queries against an XQConnection implicitly closes previous XQResultSequences spawned from the connection. One XQJ TCK test actually goes against the XQJ specification by trying to access an obsolete forward only XQResultSequence. Essentially the XQJ 1.0 Specification is incompatible with the XQJ TCK in this instance and this implementation chooses to conform to the XQJ 1.0 Specification.

Compliance Definition Statement

The compliance definition for XQJ (section 3 of the specification) requires a statement of how all aspects of the specification that are implementation-defined have been implemented. The following information provides this statement for the BaseX XQJ implementation.

The class name of the XQDataSource implementation

net.xqj.basex.BaseXXQDataSource

All properties defined on the XQDataSource implementation

The following properties are defined: serverName, port, user, password, databaseName and description.

The syntax and semantics for commands, assuming executing commands through XQExpression is supported.

BaseX UPDATE language is supported.

Is cancelling of query execution supported?

No, Query execution cannot be cancelled.

The default and supported values for each parameter described in XQuery Serialization

Although this is implementation-defined, the test suite makes some assumptions and these have been followed. The defaults are: byte-order-mark="no" cdata-section-elements="" doctype-public=null doctype-system=null encoding="utf-8" indent="no" media-type="application/xml" method="xml" normalization-form="none" omit-xml-declaration="yes" standalone="omit" undeclare-prefixes="no" use-character-maps="" version="1.0".

Additional StAX or SAX event types being reported, beside the event types documented in [the] specification

None.

Support for XDM instances and types based on user-defined schema types

Not supported.

The semantics with respect to node identity, document order, and full node context, when a node is bound to an external variable.

Node identity, document order and full node context is lost when binding a node to an external variable.

Is login timeout supported?

No.

Are transactions supported?

No.

Behaviour of the getNodeUri() method, defined on XQItemAccessor, for other than document nodes.

Will return a blank string unless the item is of type attribute() or xs:QName, in which case the value will be the namespace URI part of the items expanded-QName.

Behaviour of the getTypeName() method, defined on XQItemType, for anonymous types.

Anonymous types are returned as xs:untypedAtomic

Behaviour of the getSchemaURI() method, defined on XQItemType

It will always return null.

Behaviour of the createItemFromDocument() methods, defined on XQDataFactory, if the specified value is not a well-formed XML document.

An exception is thrown.

Behaviour of the bindDocument methods, defined on XQDynamicContext, if the specified value is not a well-formed XML document.

An exception is thrown.

The error codes, reported through XQQueryException, in addition to the standard error codes listed in [XQuery] and its associated specifications.

BaseX error codes can be found at http://docs.basex.org/wiki/XQuery_Errors

Downloading and Using BaseX XQJ

BaseX XQJ is available for download at http://xqj.net/basex.

This chapter outlines the basics of setting up BaseX XQJ.

BaseX XQJ Download package explained

The distribution of BaseX XQJ has the following directory structure:

Document or Directory	Description
`/lib/basex-xqj-1.1.0.jar`	BaseX XQJ binaries. Depends on /lib/xqj-api-1.0.jar and /lib/xqj2-0.0.2.jar
`/lib/xqj-api-1.0.jar`	The XQJ (JSR 225) Interface classes.
`/lib/xqj2-0.0.2.jar`	XQJ2 extensions, this jar depends on /lib/xqj-api-1.0.jar
`/lib/basex-xqj-examples-1.1.0.jar`	Compiled binaries of BaseX XQJ example applications.
`/LICENCE.txt`	The licence that is shipped with BaseX XQJ.
`/README.txt`	The basic overview of the download package.
`/src/*`	Contains the source files for all BaseX XQJ example applications.
`/doc/documentation.pdf`	BaseX XQJ Documentation.
`/doc/javadoc/*`	Contains the JavaDoc for JSR 225: XQuery API for Java™.

BaseX XQJ and Maven

The Maven2 Repository for all XQJ Drivers provided by XQJ.NET is located here:

```
http://xqj.net/maven
```

Setting up BaseX XQJ within the <oXygen/> XML Editor

As BaseX XQJ conforms to the XQJ Specification, it can be used within the <oXygen/> XML Editor as a means of submitting XQuery expressions to a BaseX XML Database instance and processing the results.

For full details on how to integrate BaseX XQJ with the <oXygen/> XML Editor, please see the Transforming XML Documents Using XQuery [2] section within the <oXygen/> XML Editor User Manual.

Using the Example Applications

The downloadable BaseX XQJ package includes example source files which make use of BaseX XQJ. These programs can be modified and used as a starting point for your application.

The downloadable BaseX XQJ package also includes the binaries of the example applications for ease of use.

Setting Up Your Environment

Before running the example applications, both the BaseX XQJ environment and the BaseX XML Server environment must be set up correctly.

2 Transforming XML Documents Using XQuery - http://www.oxygenxml.com/doc/ug-oxygen/transforming-xml-documents-using-xquery.html#xqj-transformer-support

Setting Up Your BaseX XML Server Environment

Before running the example applications, the following must be in place:

1 **A BaseX XML Database**. Either install the BaseX XML Database locally or make sure that the examples can connect to a BaseX XML Database else where on the network. For details on installing the BaseX XML Database see the BaseX Installation Guide.

Setting Up Your Java Environment

Java 1.5 or later must be installed on the client machine. If you have a version of Java prior to 1.5 or do not have any version of Java installed, download and install the latest version of Java from http://java.sun.com/.

The BaseX XML Database itself may require a later version of Java installed. It is recommended to use the latest version of Java.

Example Applications

This section contains information on the example applications, a brief overview of what each application does and how to run the application. Make sure you have the nessacary environment to run the applications.

Hello World Application

This is the simplest of the example applications. This applications creates a connection to a BaseX XML Database then submits the XQuery expression `"Hello World"`. BaseX returns the result of the expression where the application then prints this to the standard output stream.

The following is a sample command which will execute the Hello World Application from the command line:

```
java -cp c:/bxqj/lib/basex-xqj-1.1.0.jar; ^
  c:/bxqj/lib/basex-xqj-examples-1.1.0.jar; ^
  c:/bxqj/lib/xqj2-0.0.2.jar; ^
  c:/bxqj/lib/xqj-api-1.0.jar ^
  net.xqj.basex.examples.HelloWorld ^
  localhost 1984 bx-user bx-pass
```

The runtime parameters given in the previous sample state that the application should connect to a BaseX Server running on "localhost" which is listening on port "1984" and authenticates with the server via the username "bx-user" and password "bx-pass".

Binding Variables Application

This application creates a connection to a BaseX Server then submits an XQuery expression to BaseX. Unlike the previous example, the XQuery expression declares external variables which means they must be set from the Java environment instead of within the XQuery expression itself.

In this instance, a String and an XML Document are bound to variables within the XQuery from Java. After binding the variables, the XQuery is submitted to the BaseX server. The server then returns the two values, which are written to the standard output stream.

The following is a sample command which will execute the Binding Variables Application from the command line:

```
java -cp c:/bxqj/lib/basex-xqj-1.1.0.jar; ^
  c:/bxqj/lib/basex-xqj-examples-1.1.0.jar; ^
  c:/bxqj/lib/xqj2-0.0.2.jar; ^
  c:/bxqj/lib/xqj-api-1.0.jar ^
  net.xqj.basex.examples.BindingVariables ^
  127.0.0.1 1984 bx-user bx-pass
```

The runtime parameters given in the previous sample state that the application should connect to a BaseX Server running on "localhost" which is listening on port "1984" and authenticates with the server via the username "bx-user" and password "bx-pas".

Document Loader Application

This application creates a connection to a BaseX Server and uploads a local XML Document from the hard-disk to BaseX.

This will require you to create a BaseX Database, please refer to the BaseX XML Database Documentation for information regarding creating Databases.

Create a Database called TEST

The following is a sample command which will execute the Document Loader Application from the command line:

```
java -cp c:/bxqj/lib/basex-xqj-1.1.0.jar; ^
  c:/bxqj/lib/basex-xqj-examples-1.1.0.jar; ^
  c:/bxqj/lib/xqj2-0.0.2.jar; ^
  c:/bxqj/lib/xqj-api-1.0.jar ^
  net.xqj.basex.examples.DocumentLoader ^
  localhost 1984 bx-user bx-pass TEST c:/path/to/document.xml
```

The runtime parameters given in the previous sample state that the application should connect to a BaseX Server running on "localhost" which is listening on port "1984" and authenticates with the server via the username "bx-user" and password "bx-pass", additionally this time passing the name of the default context database "TEST" and the URI of the document which is to be loaded into BaseX "c:/path/to/document.xml".

Executing Commands Application

This application shows how one can execute proprietary BaseX commands, such as creating and modifying Databases, indexes and so on. A full list of BaseX Commands can be found at http://docs.basex.org/wiki/Commands.

Some BaseX Commands have output, such as querying about current Users or Databases, the output of these commands will be sent to the XQJ Driver's Log Writer.

The following is a sample command which will execute the Example Commands Application from the command line:

```
java -cp c:/bxqj/lib/basex-xqj-1.1.0.jar; ^
  c:/bxqj/lib/basex-xqj-examples-1.1.0.jar; ^
  c:/bxqj/lib/xqj2-0.0.2.jar; ^
  c:/bxqj/lib/xqj-api-1.0.jar ^
  net.xqj.basex.examples.ExampleCommands ^
  localhost 1984 bx-user bx-pass
```

The runtime parameters given in the previous sample state that the application should connect to a BaseX Server running on "localhost" which is listening on port "1984" and authenticates with the server via the username "bx-user" and password "bx-pass".

Invoking Module Functions Application

This application shows how to invoke stored XQuery Library Module Functions from Java "as if" they were regular Java methods. [3]

In order to run this application you must first load the "`example.xqm`" XQuery Library Module into BaseX which is included in the `/src/net/xqj/basex/examples/functions` directory.

To install the XQuery Library Module, follow the next two steps

3 Building Bridges from Java to XQuery, XML Prague 2012, Charles Foster http://www.xmlprague.cz/2012/files/xmlprague-2012-proceedings.pdf#page=197

1. Run the `basexclient` command line application
2. Replacing FULL-PATH-TO as necessary, execute the following command:

```
REPO INSTALL c:\FULL-PATH-TO\src\net\xqj\basex\examples\functions\example.xqm
```

The following is a sample command which will execute the Invoke Module Functions example application from the command line:

```
java -cp c:/bxqj/lib/basex-xqj-1.1.0.jar; ^
  c:/bxqj/lib/basex-xqj-examples-1.1.0.jar; ^
  c:/bxqj/lib/xqj2-0.0.2.jar; ^
  c:/bxqj/lib/xqj-api-1.0.jar ^
  net.xqj.basex.examples.functions.InvokingModuleFunctions ^
  localhost 1984 bx-user bx-pass
```

The runtime parameters given in the previous sample state that the application should connect to a BaseX Server running on "localhost" which is listening on port "1984" and authenticates with the server via the username "bx-user" and password "bx-pass".

Technical Support

To raise bugs or issues, please visit http://xqj.net/issues. Serious bugs will be fixed very quickly.

When raising bugs or issues, please give clear and concise explanations of how to recreate the problem with one or more **isolated** examples.

Known Limitations

- Does not support the ability to define transactions as read-only.
- Does not support XQuery 3.0 Datatypes or BaseX proprietary types such as basex:binary
- Can not bind to XQuery 3.0 external variables which have default values.

All of these limitions will be addressed in future versions of BaseX XQJ.