MarkLogic XQJ API Documentation

Version: 1.0.0, 1st September 2013

http://www.xqj.net/marklogic

Introduction to the MarkLogic XQJ API

The MarkLogic XQJ API is an interface which can communicate with a MarkLogic XML Server. Importantly the MarkLogic XQJ API 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 the MarkLogic XQJ API

The MarkLogic XQJ API is used to communicate with a MarkLogic XML Server 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. Neither the MarkLogic XML Content Connector For Java (XCC/J) or XDBC libraries implement any such industry wide standard. The MarkLogic XQJ API is designed primarily as an "industry standard interface" to MarkLogic XML Server from the Java environment. The MarkLogic XQJ API is not designed to replace the MarkLogic XCC/J or XDBC libraries. Rather it is designed as an alternative approach to communicate with a MarkLogic XML Server from a Java environment.

The MarkLogic XQJ API was built from the ground up and is not built upon XCC/J or XDBC libraries.

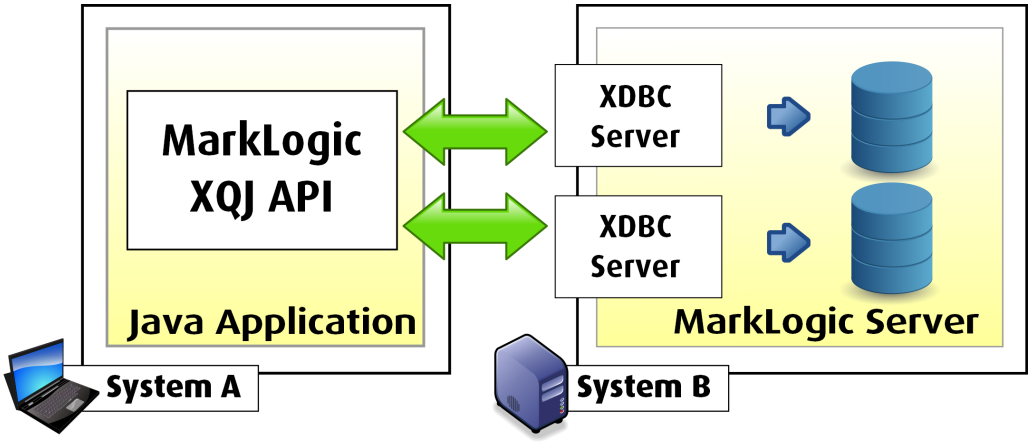
MarkLogic XQJ API communicates with an XDBC Server

The MarkLogic XQJ API requires an XDBC Server to be configured on a MarkLogic XML Server.

An XDBC Server provides a low-level network based interface, allowing external programming environments (outside of MarkLogic) to communicate with a MarkLogic XML Server. A MarkLogic XQJ API application connects to an XDBC Server running on a port on a system running MarkLogic XML Server. MarkLogic XQJ API communicates with MarkLogic by submitting XQuery statements over the wire and processing the results. MarkLogic XQJ API enabled applications can be written either as standalone or as part of a application server environment.

Client-Server Architecture

MarkLogic XQJ API communicates with a MarkLogic XML Server via a client-server architecture, where the MarkLogic XQJ API acts as the client and MarkLogic XML Server acts as the server. The following diagram illustrates this concept at a high level:



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

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

In the diagram, the MarkLogic XQJ API is used as part of a Java Application on System A. The application submits XQuery statements to a on System B by connecting to XDBC Servers configured on the server. The MarkLogic XML Server evaluates these queries against a Database which the server instance manages and returns results back to the client for processing.

Connection Pooling

The MarkLogic XQJ API automatically pools connections. There is no need to write a custom connection pooling mechanism.

It is not essential, but recommended that the `close()` method is called on `XQConnection` objects once they have been finished with.

API and Other Documentation

This section provides an introduction to the MarkLogic XQJ API library. For more detailed API documentation for the MarkLogic XQJ API and the MarkLogic XML Server or to find out how to configure XDBC Servers, please use the following resources:

Java API documentation
: http://www.xqj.net/javadoc

MarkLogic XML Server Application Developer's Guide
: http://developer.marklogic.com/pubs/6.0/books/dev_guide.pdf

MarkLogic Server Administrator's Guide
: http://developer.marklogic.com/pubs/6.0/books/admin.pdf

MarkLogic Built-In and Module Functions Reference
: http://developer.marklogic.com/pubs/6.0/apidocs/All.html

MarkLogic XQJ API (Server Requirements)

MarkLogic XQJ API was built and tested against MarkLogic XML Server 6.0. MarkLogic XQJ API may work with previous versions of MarkLogic XML Server, but these versions have not been tested.

MarkLogic XQJ API (Client Requirements)

- Requires Java 1.5 or later

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

The MarkLogic XQJ API binary jar file has no dependencies other than the XQJ/XQJ2 API interfaces.

Programming with the MarkLogic XQJ API

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

Configuring an XDBC Server

Log into the HTTP Admin Interface to create an XDBC Server. You will need to specify

- A name
- A port
- A database to access (content repository)

For detailed instructions which describe how to do this, please refer to the Administrator's Guide on the MarkLogic developer website.

It is essential that an XDBC Server is setup on the MarkLogic XML Server because the MarkLogic XQJ API connects to this.

XQJ Connections

- XQConnection objects can be generated from an XQDataSource instance
- The MarkLogic XQJ API XQDataSource implementation is `net.xqj.marklogic.MarkLogicXQDataSource`

To create an XQDataSource instance, write the following

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

An XQDataSource can be used repeatedly to generate new XQConnection objects. Before creating XQConnection objects, the XQDataSource must be given information about the XDBC Server which new XQConnections will connect to.

A MarkLogic XQDataSource instance must be given at least a host and a port, the following example tells an XQDataSource instance that the XDBC Server's hostname is "localhost" and that the port which the XDBC Server is listening on is "8003".

```
XQDataSource ds = new MarkLogicXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "8003");
```

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 XDBC Server and opens a XQConnection to the XDBC Server with a username of "user" and a password of "password".

```
XQDataSource ds = new MarkLogicXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "8003");

XQConnection conn = ds.getConnection("user","password");
```

Programming Basics

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

- Import the required libraries.
- Create an `XQDataSource` object via the `MarkLogicXQDataSource` implementation.
- Pass host and port information of an XDBC 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 MarkLogic XML Server 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.marklogic.MarkLogicXQDataSource;

XQDataSource ds = new MarkLogicXQDataSource();

ds.setProperty("serverName", "localhost");
ds.setProperty("port", "8003");

XQConnection xqc = ds.getConnection("user","password");

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();
```

XDBC and Conformance Modes

The MarkLogic XQJ API has two modes of communicating with an XDBC Server, XDBC Mode and Conformance Mode.

XDBC Mode

- Sends XQuery expressions to the XDBC Server unaltered.
- Builds XQJ Result Items based on the Metadata given by the XDBC protocol.
- Passes 94.1% of the XQJ Test Compatibility Kit (TCK).
- **Slightly** faster than XDBC Mode.

Conformance Mode

- All XQuery expressions sent to the XDBC Server are wrapped in another XQuery expression.
- Builds XQJ Result Items based on the result of the Query.
- Passes 100% of the XQJ Test Compatibility Kit (TCK).
- **Slightly** slower than XDBC Mode.

Changing the Mode

To change the Mode the MarkLogic XQJ API communicates with XDBC Servers, set a property on an XQDataSource object. The property name to set is "mode" and the value can be either "xdbc" for XDBC Mode or "conformance" for Conformance Mode.

The following example creates a new XQDataSource, then tells the XQDataSource instance that all new XQConnection objects it creates must communicate with an XDBC Server in Conformance Mode.

```
XQDataSource ds = new MarkLogicXQDataSource();

ds.setProperty("mode", "conformance");
```

When the Mode is changed, all previously generated XQConnection objects will continue to function in the same Mode they had when they were first created. Changing the Mode property on an XQDataSource will only affect new XQConnection objects.

XQJ Conformance

The XQJ API 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.

The following table shows the results of running the XQJ TCK against the MarkLogic XQJ API implementation, both Conformance and XDBC Modes have been tested separately.

XQJ TestCase	Conformance Mode	XDBC Mode
SignatureTest	PASS (1/1)	PASS (1/1)
XQCancelledExceptionTest	PASS (1/1)	PASS (1/1)
XQConnectionTest	PASS (17/17)	PASS (17/17)
XQDataFactoryTest	PASS (33/33)	PASS (33/33)
XQDataSourceTest	PASS (11/11)	PASS (11/11)
XQDynamicContextTest	PASS (21/21)	FAIL (18/21)
XQExceptionTest	PASS (5/5)	PASS (5/5)
XQExpressionTest	PASS (9/9)	PASS (9/9)
XQItemAccessorTest	PASS (20/20)	FAIL (13/20)
XQItemTest	PASS (2/2)	PASS (2/2)
XQItemTypeTest	PASS (13/13)	PASS (13/13)
XQMetaDataTest	PASS (25/25)	PASS (25/25)
XQPreparedExpressionTest	PASS (9/9)	PASS (9/9)
XQQueryExceptionTest	PASS (12/12)	PASS (12/12)
XQResultItemTest	PASS (1/1)	PASS (1/1)
XQResultSequenceTest	PASS (1/1)	PASS (1/1)
XQSequenceTest	PASS (25/25)	FAIL (20/25)
XQSequenceTypeTest	PASS (3/3)	PASS (3/3)
XQStackTraceElementTest	PASS (7/7)	PASS (7/7)
XQStackTraceVariableTest	PASS (3/3)	PASS (3/3)
XQStaticContextTest	PASS (35/35)	PASS (35/35)
Pass Percentage	100%	94.1%

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 MarkLogic XQJ API implementation.

The class name of the XQDataSource implementation

net.xqj.marklogic.MarkLogicXQDataSource

All properties defined on the XQDataSource implementation

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

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

No commands are supported (only XQuery expressions).

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

MarkLogic XML Server can execute Schema-Aware XQuery and supports user-defined schema types, when the MarkLogic XQJ API receives these result items whose type is user-defined, the type is cast to the originally standard XDM form from which it was extended from.

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, not at present.

Are transactions supported?

Yes.

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

- In XDBC Mode, this will always return a blank string.
- In Conformance Mode, it 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 item's 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.

Far too many to list here and the list is continually growing. Refer to the XML files contained within the Messages directory within a MarkLogic XML Server installation directory.

Downloading and Using the MarkLogic XQJ API

The MarkLogic XQJ API is available for download at http://www.xqj.net/marklogic.

This chapter outlines the basics of setting up the MarkLogic XQJ API.

MarkLogic XQJ API Download package explained

The distribution of the MarkLogic XQJ API has the following directory structure:

Document or Directory	Description
`/lib/marklogic-xqj-1.0.0.jar`	The MarkLogic XQJ binaries. Depends on /lib/xqjapi.jar and /lib/xqj2-0.2.0.jar
`/lib/xqjapi.jar`	The XQJ (JSR 225) Interface classes.
`/lib/xqj2-0.2.0.jar`	XQJ2 extensions, this jar depends on /lib/xqjapi.jar
`/lib/marklogic-xqj-examples.jar`	Compiled binaries of the MarkLogic XQJ API example applications.
`/LICENCE.txt`	The licence that is shipped with the MarkLogic XQJ API.
`/README.txt`	The basic overview of the download package.
`/src/*`	Contains the source files for all the MarkLogic XQJ API example applications.
`/doc/documentation.pdf`	The MarkLogic XQJ API Documentation.
`/doc/javadoc/*`	Contains the JavaDoc for JSR 225: XQuery API for Java™.

Setting up the MarkLogic XQJ API within the <oXygen/> XML Editor

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

For full details on how to integrate the MarkLogic XQJ API 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 MarkLogic XQJ API package includes example source files which make use of the MarkLogic XQJ API. These programs can be modified and used as a starting point for your application.

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

Setting Up Your Environment

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

Setting Up Your MarkLogic XML Server Environment

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

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

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

2 **A XDBC Server**. Make sure the instance of MarkLogic XML Server has an XDBC Server. You can set up an XDBC Server using the MarkLogic HTTP Admin Interface. Take a look at the Server Administrator's Guide for details on how to create an XDBC Server.

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/.

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 MarkLogic XDBC Server then submits the XQuery expression `"Hello World"`. MarkLogic 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:/mlxqj/lib/marklogic-xqj-1.0.0.jar;c:/mlxqj/lib/marklogic-xqj-examples.jar"
  net.xqj.marklogic.examples.HelloWorld
  127.0.0.1 8003 username password
```

The runtime parameters given in the previous sample state that the application should connect to a MarkLogic XDBC Server running on "127.0.0.1" (the local host) which is listening on port "8003" and authenticates with the server via the username "username" and password "password".

Binding Variables Application

This application creates a connection to a MarkLogic XDBC Server then submits an XQuery expression to MarkLogic. 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 MarkLogic 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:/mlxqj/lib/marklogic-xqj-1.0.0.jar;c:/mlxqj/lib/marklogic-xqj-examples.jar"
  net.xqj.marklogic.examples.BindingVariables
  127.0.0.1 8003 username password
```

The runtime parameters given in the previous sample state that the application should connect to a MarkLogic XDBC Server running on "127.0.0.1" (the local host) which is listening on port "8003" and authenticates with the server via the username "username" and password "password".

Document Loader Application

This application creates a connection to a MarkLogic XDBC Server and uploads a local in-memory XML Document, the URI of the XML document within MarkLogic will be `/document.xml` as this is hard-coded in the source code for simplicity.

Each MarkLogic XDBC Server is configured with a default storage Database, by default the MarkLogic XQJ API will always read from and and write to this database.

To run this application successfully, you will need a well-formed XML file to insert into MarkLogic on the local disk.

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

```
java -cp
"c:/mlxqj/lib/marklogic-xqj-1.0.0.jar;c:/mlxqj/lib/marklogic-xqj-examples.jar;c:/mlxqj/lib/xqj2-0.2.0.jar"
  net.xqj.marklogic.examples.DocumentLoader
  127.0.0.1 8003 username password
```

The runtime parameters given in the previous sample state that the application should connect to a MarkLogic XDBC Server running on "127.0.0.1" (the local host) which is listening on port "8003" and authenticates with the server via the username "username" and password "password".

Technical Support

Comments, bug finds and suggestions are most welcome.

Please visit http://www.xqj.net to report all comments, bugs and suggestions.

Known Limitations

- Currently, does not process MarkLogic binary content or MarkLogic proprietary data types.
- Currently, does not support connections over secure http (HTTPS).
- Currently, the XQuery "1.0-ml" language is not supported, only the official XQuery "1.0" language.

All of these limitions will be addressed in future versions of the MarkLogic XQJ API.