Xindice 1.1 Administration Guide

$Revision: 1.16 $

by Kimbro Staken, Gianugo Rabellino

NOTICE:

1. Database Administration

Database administration of Xindice is accomplished from the command line using the `xindice` command. This command allows you to view and alter the database configuration on the fly on a running system.

1.1. Managing Collections

1.1.1. Adding a Collection

Adds a collection named products under the collection /db/data.

```
xindice add_collection -c /db/data -n products
```

1.1.2. Deleting a Collection

Deletes the collection named products from the collection /db/data.

```
xindice delete_collection -c /db/data/products
```

1.1.3. Listing the Collections

This will display a list of all child collections under the collection /db/data

```
xindice list_collections -c /db/data
```

1.2. Managing Indexes

The Xindice indexing system allows you to define indexes to speed performance on commonly used XPath queries. If no indexes are defined you can still execute queries but performance will suffer because the query engine will need to scan the entire collection to create the result node-set.

Indexes can be added using the `xindice` command.

1.2.1. Adding an Index

Using this simple XML file you might want to index the product_id element because searches for products by product_id are common.

```
<?xml version="1.0"?>
<product>
  <product_id>120320</product_id>
  <description>Glazed Ham</description>
</product>
```

This can be accomplished by running the following command. This will create an index named idindex on all product_id elements in the collection /db/data/catalog.

```
xindice add_indexer -c /db/data/catalog -n idindex -p product_id
```

Once this is done the query engine will now use this index to help resolve XPath queries that involve restriction on the value of the product_id element.

The -p parameter to the command specifies the pattern to use in the index. These patterns are used by the Indexing system to determine best-fit and match-based Indexers for queries and index updating. The pattern used MUST resemble the following scheme.

```
Pattern      Description
===========  ==================================================
elem         The value of the named element
elem@attr    The value of the attribute for the named element
*            The value for all elements
*@attr       The value of the named attribute for all elements
elem@*       The value of all attributes for the named element
*@*          The value of all attributes for all elements
```

Note: In order to index a namespace other than the default namespace, you must prepend your pattern components with a URI placed in square brackets. Example:

```
[http://www.world.org/People]person
*@[http://www.world.org/People]id
[http://www.world.org/People]person@[http://www.world.org/People]id
```

Do not include a prefix in these patterns, as the indexing system, like most Namespace processing applications, processes namespaced elements and attributes independently of the prefix that is used.

1.2.2. Indexing both Elements and Attributes

Because the patterns recognize either an element or an attribute, and not both, in order to index all element and attribute values in a collection, you'd have to create two index entries. The * pattern will index all elements and the *@* pattern will index all attributes of all elements.

```
xindice add_collection_indexer -c /db/data/catalog -n idindex -p '*'
xindice add_collection_indexer -c /db/data/catalog -n idindex -p '*@*'
```

Excessive use of wildcard indexes can adversely affect the performance of the indexing system. Best practice would be to use specific element or attribute indexes whenever possible, and only define wildcard indexes when it is absolutely necessary.

2. Server Administration

2.1. Installing the Server

Starting from 1.1, Xindice is not a standalone server anymore. The server functions are now based on your favourite Servlet 2.2 (or 2.3) compliant Application Server. Xindice has been tested and proven to work under both Tomcat and Jetty, but there is no particular reason to expect malfunctions under other application servers.

Installation is then straightforward: just deploy the Xindice WAR file (`xindice-1.1b3.war`) into your favourite application server and you're ready to go. There are only two minor points to be aware of:

- The Xindice XML-RPC endpoint is configured in the client as `http://anyserver:anyport/xindice/`. This means that it's strongly advisable to deploy the Xindice WAR file under a `xindice` context. This can be easily accomplished under Tomcat by simply renaming the WAR file to `xindice.war` or (in Tomcat 4.1.x) by copying the file `dist/xindice-1.1b3.xml` under the `$TOMCAT_HOME/webapps` directory. Note that under some Tomcat versions you will need to start twice the server the first time so that Tomcat can configure itself properly.
- You probably want to edit the Xindice configuration file that resides under `/WEB-INF/system.xml`. This file configures, among others, the physical location of the database. By default, your data will be under

`[your_unpacked_war_location]/WEB-INF/db`, which might not be a good idea for many users: leaving the database as is will mean data loss if an upgrade takes place inadvertently, since the directory will be overwritten. Also, if your application server is not unpacking WARs, Xindice won't be able to start.

Having the server packaged as a webapp means also that starting and stopping Xindice is "just" a matter of starting/stopping the application server.

2.2. Starting the Server

Assuming that you have installed Xindice under Tomcat, and that you have TOMCAT_HOME environment variable pointing to Tomcat installation directory.

2.2.1. Starting the Server on UNIX

```
cd $TOMCAT_HOME/bin
./startup.sh
```

2.2.2. Starting the Server on Windows

```
cd %TOMCAT_HOME%\bin
startup.bat
```

2.3. Stopping the Server

To stop Xindice server, you just stop application server. Assuming that you are using Tomcat.

2.3.1. Stopping the Server on UNIX

```
cd $TOMCAT_HOME/bin
./shutdown.sh
```

2.3.2. Stopping the Server on Windows

```
cd %TOMCAT_HOME%\bin
shutdown.bat
```

2.4. Backing up Your Data

2.4.1. Backing up the server

Just shutdown the application server and copy the db directory structure somewhere else, e.g. using Tomcat and the server version of Xindice with the default configuration:

```
catalina.sh stop
cd $TOMCAT_HOME/webapps/xindice/WEB-INF
cp -pr db /backup/db
catalina.sh start
```

2.4.2. Restoring the Data

Restoring the data is simply removing the current database and reversing the backup process. Again, using Tomcat, this will be something like:

```
catalina.sh stop
cd $TOMCAT_HOME/webapps/xindice/WEB-INF
rm -rf db
cp -pr /backup/db db
catalina.sh start
```

2.5. Exporting the Contents of the Database

Xindice includes tools to export data to a directory hierarchy and to also import data from a directory hierarchy. Each directory in the hierachy corresponds to a collection in Xindice. Each XML document is stored in a separate file named with the key from the database.

2.5.1. Exporting the database

This example assumes that the Xindice/bin directory is in your path.

```
xindice export -c /db/root -f /path/to/data
```

The entire contents of the collection /db/root will be exported to the directory /path/to/data.

2.5.2. Importing the database

This example assumes that the Xindice/bin directory is in your path.

```
xindice import -c /db  -f /path/to/data/root
```

Each directory under /path/to/data will be used to create a collection and all XML documents

in the hierarchy will be imported in to the database. You can also restrict the documents that are imported by adding -i and the extension of the files you want to import.