Ryan Liu
CSS 499
Winter 2004
Independent Research: UW Messenger

# Xindice: XML Database

## I. Introduction:

The UW messenger research group has chosen Xindice as the database implementation to store all XML files: resources, requests, and users. It is currently installed on the perseus, and several java files have been implemented by Eric Nelson and Ryan Liu. ResourceAgent and outside users can interact with Xindice through XCollection..

## II. Useful Links:

a. http://xml.apache.org/xindice/guide-developer.html (examples of xindice implementation and helpful FAQ)
b. http://www.xmldb.org/xapi/api/overview-summary.html (specification of all Xindice classes)
c. http://www.mountainstorm.com/publications/javazine.html (using java code process to execute functions for command line)
d. http://java.sun.com/j2se/1.4.2/docs/api/ (specification of class Process & class Runtime)

## III. Important Commands of Xindice:

a. start &: start up xindice db
b. xindiceadmin shutdown –c /db: shut down xindice db
c. xindice lc –c /db: list all collections in the database
d. xindice ld –c /db/resources: list all documents in a specified (resources) collection

## IV. Xindice General Behavior and Potential Problem:

After launching xindice, the *xidice.pid* file, which stores the "pid" of current running database process, will be created in the XINDICE_HOME/config/ directory. It will check the existence of *xindice.pid* before executint each 'start &' command. Any instance of class 'collection' can use 'close' function to close collection which basically removes the *xindice.pid* file from the 'config' directory. This is a significant consideration when the project is close to finish because there is no guarantee that xindice will be shut down properly each time.

Also, we have noticed that it takes some time for xindice to start up and which will cause problem if our program try to execute some xindice command without it starting up completely. Although our current constructor

will wait for its start up completion, this behavior should take into account if any change to the constructor in the future.

## V. XCollection (interface between agents and Xindice db) :

a. Design:

The UW Messenger project requires three types of XML file: users, resources, and requests. Since these files have mostly similar functionalities, we have created a super class XCollection for the general functionalities: insert, delete, retrieve, query and etc. UsersXCollection, ResourcesXCollection, and RequestsXColletion are the three sub classes derived from the XCollection. Each sub-class interacts with its corresponding collection in the xindice.

In order to prevent any multiple readers and writers working on the same file at the same time, each sub-class of XCollection can instantiate only one instance each time for each collection. This will solve the potential multi-editing problem.

When first instantiate an instance of XCollection, the instance will call the super's constructor to start up xindice before future executing any commands. Technically, the last instance of XCollection should call the destructor before shutting down xindice, but we haven't decided whether this should be implemented in the ResourceAgent or XCollection.

b. Specification:

**i. *Classes*:**

Each sub-class has its own functionality to interact with corresponding collection in the Xindice.

1. XCollection: Super Class
2. ResourcesXCollection: Sub Class of XCollection
3. RequestsXCollection: Sub Class of XCollection
4. UsersXCollection: Sub Class of XCollection

**ii. *Functions*:**

| | |
|---|---|
| Private String name | Name of the current collection |
| Private Collection col | Current working collection |
| Constructor ( String colname ) | Start up db if necessary, and initialize both private members name with colname and col for current working collection |
| GetInstance ( ) | If current collection's instance has not yet been instantiated than call super's constructor to initialize the instance of current collection |

| | |
|---|---|
| Void Insert ( String args[]) | Inset multiple XML files to the corresponding collection. The passing string array stores the paths to each XML file |
| Void Insert ( String element) | Insert a new XML file with a given path to the corresponding collection |
| String ReadFileFromDisk ( String fileName) | Get the content of the inserting file from the given path, and return the content to the calling function (Insert) |
| String GetfileName(String path) | Parsing file name from a given path to the inserting file return the desired file name to the calling (insert) function |
| Boolean[] Delete ( String args[]) | Function to delete multiple XML files specified by the input 'args' array. Return boolean array, and each element represents corresponding remove condition. True for delete successfully, false otherwise |
| Boolean[] Delete ( String element) | Delete a XML file, file name specified by element, from the current collection and return true if remove successfully, false otherwise |
| String Retrieve ( String args[]) | Function to retrieve multiple XML files from the database Display the content of each file if found corresponding files, error message otherwise. |
| String Retrieve ( String element) | Function to retrieve a XML file with given file name ‘file’from the database. Display the content the file if found, error message otherwise. |
| Destructor( ) | Call the ‘close’ function to close the current collection before shut down the db. |
| Vector Query(xpath) | Searching for specific data specified by "xpath" in each XML file in the current collection. Searching algorithm is based on Xpath service. Return the best matched file set to the calling function (file name only) |
| Update(String field, String value) | The ideal of “Update” function will allow user to change specific attribute of one or multiple XML files in the database by using Xpath service. Although there are more provided functionalities by Xindice, we might not use all of them in order to keep the consistency format of each collection files. |