Course Design, Representation and Browser for Web Based Education

KUNAL CHAWLA
Department of Information Technology
Indian Institute of Information Technology
Allahabad, Uttar Pradesh
INDIA

Abstract: - Web based Education is always looked upon as a fascinating field in Distant Education. A lot have been discussed in using computer technology in the delivery of education. The paper deals with the development of a protocol for the course or learning material in a XML format which can be saved in XML databases. The paper also showcases the development of a module which provides a browser for displaying the learning components and simplifying the e-education for the learners. A learning component can be a text or image or a video clip etc. Apart from the display of learning components, the module provides additional features which can make the e-learning more interactive and easier. The module also involves a tool for providing communication between the learners and the teachers or professors through a messenger application. Therefore the module tries to provide an effective distance learning tool by covering few of the issues of distant learning

Key-words: - Distance Learning, Learning Management Systems, Learning Materials, Computer Assisted

1 Introduction

This distance education tool for display and representation of learning material not just aims at providing an interface for display of various learning components but also provides an interactive application for overall effective learning of a learner. It also showcases a design for the learning material or course file in XML. The course material exists as XML files along with the various learning components and thus taking the advantage that XML provides. The learning components may vary from simple text components like rtf files or images to audio/video files. Along with that the application provides the user with a facility to save notes like in a class room lecture environment. Also the user can bookmark the important pages and can visit them in course of learning. The application also provides the course material to be displayed as a slide show. There are many other features like searching for keywords in the specified course etc. which shall be covered with the course of this paper. Thus this application provides the user a more interactive browser than web browser, specifically developed for course or learning material files which exist in XML format.

From the course developer/author point of view this application provides a feature to convert the course in XML format to the course in HTML format which can be readily put on web servers and thus the course can be accessed through the most conventional ways i.e. a web page.

The course that exists in the form of XML files needs to parsed and then displayed. For parsing SAX API is used. The displayable components or the learning components are linked to the XML files. There is also a tool for communication between the application users (like students) and the course developers (like teachers) etc.. For this purpose a messenger tool is developed and added to this module. The messenger uses Java Technologies like Java RMI to interact with server, Sockets to interact with forwarding program and JDBC to interact with database and this stands as an effective communication tool. The intricacies of messenger are also covered in the course of paper.

There are few other concepts which are not implemented but are still under active discussion and their implementation and integration with the current application is a part of future work.

The comprehensive solution to web based education consists of implementation of many modules and this paper throws light on the one of such modules.

2 Major Goals

- Design of an optimized course format in XML.

- The learning components to be displayed must cover all major formats of learning components like audio, video, images, text, rtf etc.
- Providing an interactive and easy to learn environment to the user for effective learning.
- Development of a messenger application which can provide the communication not only among the users but also among the users and course authors.

3 The Course File structure

3.1 The Basic Course Structure

The Course structure is organized in a very efficient way. Accessing the main course file allows other sub-files to be accessed. The Course file (XML file) consists of many lesson file references in addition to the various attributes of the course. A Lesson (XML file) contains many screen file references in addition to various attributes of the lesson. On the same grounds screen file contains information regarding the display page and references to various components.

The learning components which can be displayed include audio, video, image, html, rtf etc. The hierarchy (Structure) of the files is as shown below

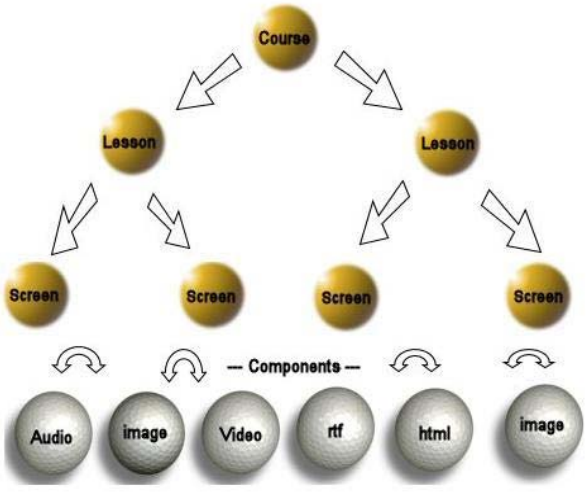

Figure 1: Hierarchy of course file structure

3.2 The Course Structure in Detail

The course file contains information about the course like the authors, copyright holder, name of the course etc. For every attribute, a tag is reserved, whose content is picked up while parsing the file.

For eg. For authors the XML syntax is

```
<authors>
        kunalChawla
</authors>
```

The Course file contains an important tag: “lesson-path” who’s content is the relative path to the lesson xml file. So, we can include many lessons in a course file and that can be attached to the course by just adding their relative paths.

The lesson file contains attributes like name of the lesson, description etc. whose content contain their respective information. The tag which links the lesson file to screen files is: “screen-path” and its role is similar to “lesson-path” in Course file.

The screen XML file contains the information regarding the frames which corresponds to a component that should be displayed. Each frame-path tag in the screen file has attributes like coordinates, length, breadth and the most important are the type and path of file which contain information regarding type like (audio/video, html, image etc) and the relative path respectively, for describing the way component should be displayed.

Following are some samples of the files of the course material.

Course1.XML

```
<course>
        <description>
                Course1
        </description>
        <authors>
                kunal Chawla
        </authors>

        <display-name>
                Java
        </display-name>
        <copyright>
                kunal Chawla
        </copyright>
        <lesson-path>
                /Lesson1.xml
        </lesson-path>
        <lesson-path>
                /Lesson2.xml
        </lesson-path>
</course>
```

Lesson1.XML

```
<lesson>
        <description>
                Lesson1
        </description>
        <display-name>
                XML
        </display-name>
        <screen-path>
                /Screen1.xml
        </screen-path>
        <screen-path>
                /Screen2.xml
        </screen-path>
</lesson>
```

Screen1.XML

```
<screen>
        <description>
                Screen1
        </description>
        <display-name>
                basics of XML
        </display-name>
        <frame-path x="0" y="10" width="137"
                height="581" type="image"
                path="/bgStrip.jpg">

        </frame-path>
        <frame-path x="147" y="10"
        width="300" height="700"
        type="rtf" path="/rtf1.rtf">
        </frame-path>
        <frame-path x="500" y="10"
        width="300" height="700"
        type="text" path="/txt1.txt">
        </frame-path>
</screen>
```

These tags which are mentioned are just the basic tags. The customized application can have as many tags as possible and a code can be developed for the browser application to decipher the information contained in those tags of the course files.

4 The Content Display Module (Browser for Course)

This module is divided into phases

1. Parsing of XML file
2. Building up the module
3. Calling various functions of the module e.g. Display, search

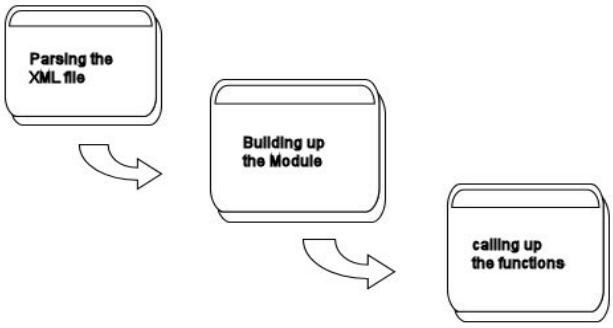

Figure 2: Phases of Content Display Module

The Course XML file to be read is passed to the module, where it is parsed using the SAX API .The SAX API is the most basic of all APIs which gives the maximum freedom to the developer, but requires maximum programming too. As the course file contains path to lesson files and lesson file contains path to screen files, the parser is programmed to parse these files too. On encountering a particular tag an event is called. So the code pertaining to a tag is appended there.

The parser can be extended by adding rules for various other tags which we want to add in XML files. Thus we can make a universal parser for the XML, by just defining tags and the execution code for that. The execution code for *-path tags works in the following way. The XML file corresponding to that particular tag is opened up and is parsed in a similar way described above and when it is parsed, the control returns back to main file.

As the tags are encountered a course model is built up and all the corresponding variables are set which occurred as tag content or tag attributes in the XML files. The model has the following structure.

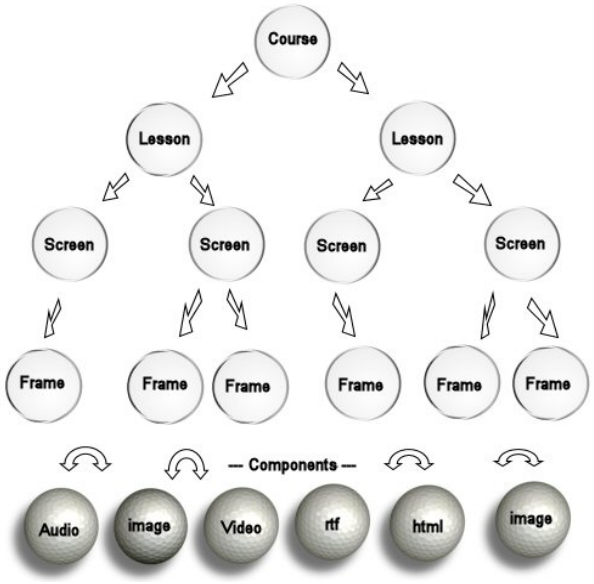

Figure 3: Hierarchy of the Course Model

The hierarchy is very clear from the above diagram. Here each node corresponds an object of the classes from {Course, Lesson, Screen, Frame, Components}. The classes of the model provide functionality and can be harnessed by the call of functions.

The functions include getting the display content, searching, building up HTML representation etc. The course file when invoked for a function which depends on other sub components: lessons and screens like getting display content builds up its own display content and calls its subcomponent functions for display which in turn asks their subcomponents for display. As a result whole tree structure is invoked and the results are combined and returned by the top most function. This feature makes the model an ideal one.

5 Features

1. Search: search searches for a keyword in the specified tag like description, type etc in the model and returns the result in form of table where each tuple references a screen.
2. Print: prints the current contents of the screen.
3. Full Screen: Full screen changes the current display settings and shows the screens in form as slide show. There are buttons for moving to next and previous screens.
4. HTML preview: This function builds up the HTML code for each of the displayable parts of the screen and then calls up windows API to display the html code in the web browser.
5. Tree representation: Each course object call its lesson objects and each lesson in turn calls its screen objects to form a JTree. This is a very efficient way of representation of the whole course and also it is useful for the user for navigation through the course.
6. Navigation :

   Following functionalities are added for navigation

   Next: for moving to the next screen

   Previous: for moving to the previous screen

   Top: for moving to the first screen

   Bottom: for moving to the last screen
7. Generate HTML book: This feature generates a separate ready to be delivered HTML document book which has a navigation tree format for course lessons, screen etc. This feature provides a mechanism to convert the XML course material to HTML format which can be readily put on web servers and the learning material can then be accessed through web pages.

   Every component is displayed on a JPanel in the application user interface. But on the HTML page, the HTML component and images are displayed by just appending their HTML script. The rtf and media components are displayed using an applet and putting the corresponding applet script to the main HTML script of the page.
8. Notes: While reading the course the reader wants to make certain notes. For this purpose, there is a facility of making notes for every screen and saving and editing them.
9. Bookmarks: You can add a screen to the list of bookmarks and the bookmarks information is saved in the workspace/bookmarks for that particular course. So every course has its set of bookmarks.
10. Communication: The messaging application discussed in the course of paper provides the instant communication between the learners and the authors of the learning material.

6 Messaging application

Communication is an integral part of learning. The Messenger Application is utilized for communication among course developers and learners. The Messenger Application includes

communication via text messages and diagrams involving basic geometrical figures like rectangle, ellipse and line.

6.1 Messenger Architecture

Whole Messenger Application involves 4 sub-modules.

1. Server
2. Database
3. Forwarding Program
4. Client

In the following paper the design of each module is illustrated and further their implementation is also explained.

The design is made in order to make the communication *reliable and fast* enough under hardware constraints.

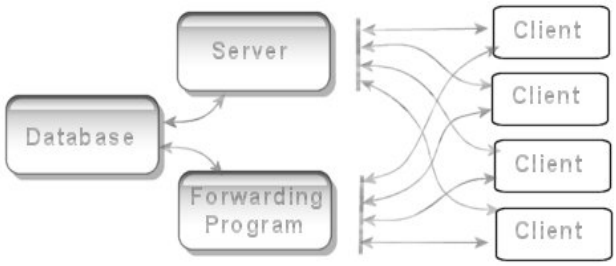

Figure 4: Structural Architecture of Messaging Application

6.2 Design and Implementation

The server, forwarding program and the client is implemented in Java. The Client communicates with a peer via the forwarding program. The clients and forwarding program interact using TCP/IP Sockets. The forwarding program interacts with the server using jdbc, while the server and the client communicate using JAVA RMI.

In order to use the application the user first needs to create an account with the Server. This includes form filling and sending that to the database. After the creation of the account user need to sign in.

In the sign in procedure user is authenticated and a session key is generated which is required in further interactions with the server and the forwarding program.

The user can add/delete/update friend's information and maintain the friend list.

The Database contains the information regarding the online status of the user, the friend list of the user, offline messages etc.

The Forwarding Program is a program which allows the clients to communicate with each other using TCP sockets. The client sends the message addressing the user to the forwarding program which is running on a machine with a known IP address.

The Forwarding program after receiving the message from a user forwards the message to the addressee. If it is unable to send the message to the required host then it returns an error message to the Client containing the same message. Client then tries to send the message using the Server, the server then tries to send the message and if it is unable to do so then it saves the message as an offline message.

The offline messages addressed to a particular client are then retrieved by the Client when he logs in. The Server basically provides a call to retrieve the offline messages and various other things. So retrieving information from the server highly depends on the design of client.

The interaction between the server, the client and the forwarding program takes place using specific protocols. Each message is coded using a specialized protocol depending on its purpose, before leaving for the destination.

Now the client design should be such that it effectively and efficiently utilizes the functionality provided by the server.

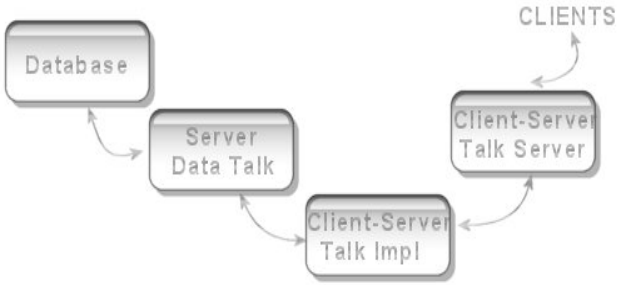

Figure 5: Structural Architecture of Server

6.2.1 Server and Database

The 'client-server talk server' receives requests from the clients and passes them an object of the 'client and server talk implementation', which provide the various functionalities to the client. The 'client and server talk implementation' in turns uses functionalities of 'server data talk' which provide an

interface between database and the ‘client server talk implementation’.

The Database consists of five tables which adhere to the requirements of the application.

The Main table maintains the user id and passwords of all the users who have an account along with general information regarding the user like name, gender, age etc. The Online_Users table has information related to all the users who are online at that time. A special key is generated each time the user logs in. This is key is unique for each session and is stored in this table along with the ipaddress of the user. There are separate tables for storing the friend lists and the offline messages of each user. The database id designed in such a way so that there is no redundancy of data and it follows all the referential integrity constraints. This keeps the information correct and up to date.

The server is designed to in order to provide all sorts of functionality to the client i.e. Creating a new account, managing of friend list, retrieval of offline messages etc.

The Database can be further expanded as per the needs of the application.

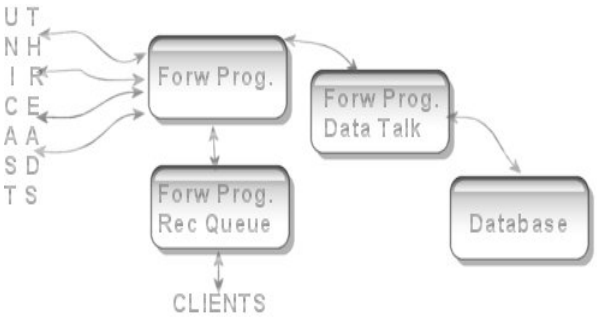

Figure 6: Structural Architecture of Forwarding Program

6.2.2 Forwarding Program

The clients and forwarding program interact using TCP/IP Sockets. The requests from the clients enter a queue maintained by the forwarding program. The forwarding program takes each element from the queue and verifies the userid and corresponding session key against those values retrieved from the database. If these values match then a unicast thread is created which forwards the message to addressed userid by finding out his ipaddress from the database. If these values are not matched then a message is passed back to sender about the invalid session key.

And if the addressed userid is offline (i.e. program was unable to fetch the ipaddress of addressee from the database) or if the ipaddress is unavailable on the network then the message is forwarded back to the sender using a unicast thread.

The forward program data talk has read only grants for the database that adds to the security.

We can have a number of such programs running on different machines at different locations. All these machines will have a universally known ipaddrress. This reduces the load on the server and hence increases the efficiency of the application.

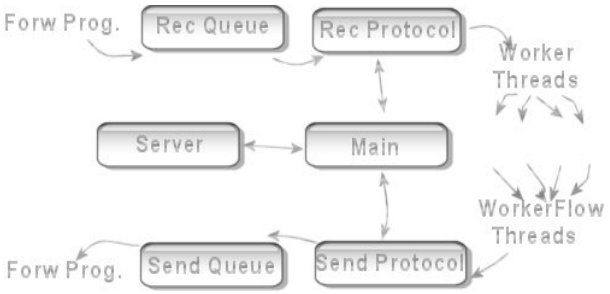

Figure 7: Structural Architecture of Client

6.2.3 Client

On the client side messages are received using TCP/IP Socket interface and are added to the receiver queue. Messages are taken from the queue one by one and are then decoded by the receiver protocol. The tasks are then assigned to Worker Threads which carry out the work desired by the corresponding protocol string(the encoded message). A task can be to perform the job of displaying them through the GUI.

Similarly on the sending side Work Flow Threads take messages from the user through the GUI. These messages are then encoded by the sender protocol in a particular format which considers the addressee, the function required etc. Then they enter the sending queue after which they are sent to the forwarding program using TCP/IP Socket interface.

The client retrieves the reference to the server object using JAVA RMI. The various functionalities provided by the server are called using this reference.

6.2.4 Protocol String

The protocol string is the data that is exchanged among the clients.

CLIENT TO CLIENT MESSAGE (via Forwarding Program)

← key →←SENDER'S ID→←DESTINATION'S ID→←CLIENT to CLIENT MESSAGE→

This is the message that is sent to the Forwarding Program. Here the "key" is padded to 6 characters, "Sender's ID" is padded to 15 characters and "Destination ID" is padded to 15 characters.

CLIENT TO CLIENT MESSAGE (after processing of forwarding program)

←TYPE→←SUB-TYPE1→←SUB-TYPE2→←SUB-TYPE3→← SENDER'S ID→←TEXT→

TYPE	:	1 CHARACTER
SUB-TYPE1	:	1 CHARACTER
SUB-TYPE2	:	1 CHARACTER
SUB-TYPE3	:	1 CHARACTER
SENDER'S ID:		15 CHARACTER
TEXT	:	500 CHARACTERS

TOTAL: 519 CHARACTERS

6.3 Messenger Features

Reliability

The forwarding program after receiving the message from a user forwards the message to addressee. If it is unable to send the message to the required host then it returns an error message to the Client containing the same message. Client then tries to send the message using the Server, the server then tries to send the message and if it is unable to do so then it saves the message as an offline message.

If the same user logs in at different machine, his previous session is destroyed.

Security

No client is allowed to access the ipaddress of any peer. Only the forwarding program's ipaddress and server's ipaddress are revealed.

Incase of forwarding program, we are using a database user with read only grants. But in case of server we have a database user with access restricted to desired amount.

This definitely adds security features to database.

Drawing Board

The drawing board provides the facility to express and convey those things which can't be conveyed by just plain text messages

7 Comparisons with other similar Applications

The conventional ways for delivery of education have been through web pages. As a result the learning process hasn't been interactive enough. By making an application like this for the e-learning that covers up all the features of web pages and provide interactivity at the same time is a better way of providing education via computer technology.

The module provides a messenger for interaction among the learners and authors of the course which is very integral part of learning.

And, the design of course in XML is an evident advantage over the course materials existing as HTML pages.

8 Conclusions

The implementation of this module for web based education was a success as it provided the tool to view course which exists in XML format. It also provided the much needed interactive application for the user. It has a capability of displaying media, image, rtf and HTML files. The feature that converts the course to HTML book which can be easily dispatched on a web server is one of the very appreciable features. An effective communication with users and course authors via Messaging Application makes this distance education tool very impressive and above par, when compared to other distant education tools

9 Future Work

For making a comprehensive application, much more work is still to be done. Some work is to be done to make the browser for course material capable of conducting online examination which can be interactive with the student and at the same time, it can assess the student in terms of time spent on a particular question, the approach taken by a student etc. i.e. addition of artificially intelligent techniques to assess the student.

The application right now covers the display of limited learning material formats. So, work has to be

done to cover the display and presentation of other learning component formats like animations in flash etc.

The work in Web Based education is quite extensive and deals with the development of many modules like NLP, Text to Speech, Document Summarizers, Mathematical Toolkit etc. The stitching together of all these separate modules to build up a comprehensive package for web based education is also a part of future work.

10 On Running the Application

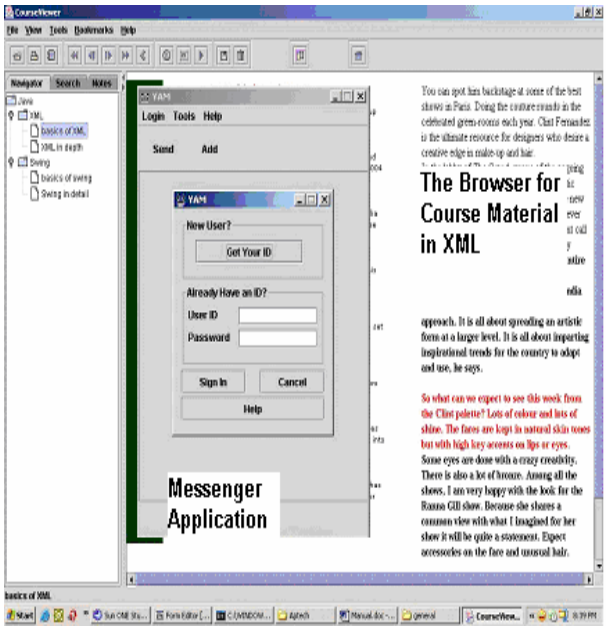

Figure 8: The look and feel of the Application

11 References

Cay S. Horstmann ,Gary Cornell "Core Java 2 " Advanced Features Vol 2 , Pearson Education Asia, Second Indian Reprint ,2000 , pp 147-312.

Cay S. Horstmann ,Gary Cornell "Core Java 2 " Vol 1 , Pearson Education Asia, Second Indian Reprint ,2000

The Java™ Web Services Tutorial Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A.

Java™ 2 SDK, Standard Edition ,Version 1.4.2