FAR: An Editing Tool for Standard Information Generation

Tao Zhang
School of Business
University of Mississippi
tzhang@olemiss.edu

H. Conrad Cunningham
Dept. of Computer Science
University of Mississippi
cunningham@cs.olemiss.edu

Jian Li
Dept. of Computer Science
University of Mississippi
leejian@olemiss.edu

Abstract

It's difficult for administrators to retrieve information from reports that are generated by different people with different computer platforms, word processors, and interpretations. In this paper, the Faculty Activity Report (FAR) in University of Mississippi is used as a case study to find a way to solve the above problem. We describe the design and implementation of an editing tool aimed at assisting the faculty members of University of Mississippi to generate their FARs. The editing tool, based on the technologies of Java Swing, XML, XSD, and JDOM, have the following characteristics: standardized input procedure, platform independence, and extensibility. We also discuss the future improvements of the editing tool.

Keyword: *Platform Independence, Extensibility, XML, JDOM, XSD*

1. Introduction

Each spring all faculty members in University of Mississippi must write a report that documents their professional activities for the previous year. The Faculty Activity Report (FAR) is prepared according to published guidelines including courses taught, papers published, grants received, and so forth. The information collected in the FAR is used by administrators in evaluation of the faculty member's performance, in preparation of other reports that combine the information across groups, and in response to various special requests for information. Because faculty members use a variety of computer platforms (Windows, Mac, UNIX, Linux) and word processors (Word, WordPerfect, Latex, etc.), and have a variety of interpretations of the guidelines, it is difficult to use the FAR source files to automatically generate other reports.

The goal of this project is to develop an editing tool that will assist faculty members in generating their FARs in a standardized format that can be more readily processed by other tools. The tool should be portable across the computing platforms that support Java Virtual Machine (JVM) and likely to be used by faculty members, be enable the reports to be written incrementally over an arbitrary period of time, and be extensible to allow the program to be changed easily if the format of the report changes.

We use the technologies of Java Swing [1] to generate the general interface. The faculty members can edit and modify their reports by working on a series of forms. Then the information stored in JTree structure can be translated into an eXtensible Markup Language (XML) document [2], and the corresponding XML Schema Definition (XSD) [3] can be used to test whether the report is valid. The valid and well-formed XML file can be converted back into a tree structure using the Java-based Document Object Model (JDOM) [4] library to allow the user to see the hierarchy of XML data and to allow queries.

The structure of this project report is as follows. The technologies that are used by the project will be described in Section 2. We present the basic design issues for our solution to the standard input procedure problem in Section 3. Section 4 will show the implementation of the technologies that are described in Section 2 with the design ideas that are stated in Section 3. The conclusion will be given in Section 5. We also listed various improvements to this project that can be undertaken in the future in Section 6.

2. Overview

This project involves fifty-six forms. Each of them is generated for the different categories in the instructions given to the faculty members to prepare their reports. It is a good practice to group the collection of cooperating classes into one location. In this paper, we use the Java package to organize all the fifty-six forms in one package.

In the project, we want to generate friendly and convenient user interfaces to allow the users to prepare their reports. We need menus, toolbars, buttons, and so on. It is desirable to allow the users to choose their own Look-and-Feel properties on their own machines. Also a tree-like data structure will be helpful for us to show the hierarchy of the instructions' categories. Java Swing [1] is an excellent choice for the project. The Java Swing architecture implements the concept of the Model-View-Controller (MVC) pattern. It improves the traditional MVC pattern by combining the Controller and View into a single User Interface (UI) delegate [5] and separates the Look and Feel object from the UI delegate to allow a pluggable Look and Feel. JTree is the one of the Swing components that is a good data structure to manage hierarchical data. JTree object is a good facility for implementing the tree structure of user interface components.

In the project, we want the final FAR to be in a uniform format, which will enable tools to be written to summarize, query, and aggregate information from many reports. Also, the FAR is a kind of structured tree-like hierarchical document and the instructions for the faculty members to prepare their reports might change unexpectedly. The eXtensible Markup Language (XML) comes to be the first choice for the project because it is a language for documents containing structured information. It is more flexible than HyperText Markup Language (HTML) because it enables the maintainers of this project to create their own tags according to the latest instructions. XML Schema Definition (XSD) is used to test whether a well-formed XML document is valid since XSD has some advantages over Document Type Definition (DTD). We generate an XSD according to the instruction for the faculty members to prepare their faculty activity reports. We can then check the well-formed XML documents against the XSD to make sure that it is valid.

To retrieve and manipulate information from structured tree-like documents easily, it is necessary to convert the well-formed XML documents into a tree-based structure. Also we want the method to use Java technologies to allow platform dependency. JDOM has all the requirements that we considered in the project. Because its specified language is Java and it converts XML documents into tree-like structure and stores the structure in memory for future uses.

3. Design

This project develops a Faculty Activity Report (FAR) editor with the following characteristics:

a) Standardized input procedure.
b) Platform independence.
c) Extensibility.

In response to the desired properties of the FAR editor, we adopt a few general approaches. The project uses the Java Swing library to create the general user interface, so that the faculty members can follow the instructions listed in forms to create their activity reports. The information is stored into the leaves of JTree structures. Faculty members also can modify the information anytime. The project uses the XML concepts and technologies. We create an XML file depending on the information that the faculty members input into the forms by using JDOM library to parse the XML files into a tree-like data structure. This makes it possible to randomly access the data in the tree.

In the project, we define the outline of the faculty activity report according to the published university guidelines for 2002 and use it as the skeleton to decide what kinds of items should be included into the forms, what kinds of elements should be included into the XSD, and also it ensures us that same items have uniform names in FAR, Forms, and XSD.

We want the faculty members to work on their reports by filling out a series of forms. The information that filled into the forms will be stored as a string with XML tag names into the JTree leaves. Root and all the JTree internal nodes store only the XML tag name. When the ConvertToXML button is clicked, the convertToXML method will get the JTree tree as input and retrieve strings from all of the nodes of the tree to generate an XML document.

The project also enables users to open and convert a well-formed XML document back into JTree by *ConvertXMLToTree.java*.

We define the XSD and use it to validate the generated XML document for the faculty activity reports using dom.ASBuilder sample.

The project addresses each of the desired characteristics as follows:

(a) To accomplish the standardized input procedure, we generate a standardized interface that includes the required categories listed in the directions for preparing the FAR. We use the JTree features of Java Swing to manipulate the hierarchical structure of the FAR. Before any information is input, all the leaves of a FAR tree are blank. Also the requirement of each category appears in the interface in HTML format. The faculty members can follow the instructions in the interface to complete their reports.

(b) To accomplish the platform independence, we use Java which is a platform independent language. So the interface that is generated by Java is also platform independent and the faculty members can edit and modify their reports on any operating system that supports the Java Virtual Machine (JVM). Different browsers can view the final report. Also JDOM, which is used in our project to parse XML, uses Java as its implementation language.

(c) To support a uniform format, we use the XML family of technologies. XML is an extensible language. We use XML Schema Definition (XSD) approach to design and implement a program for the tool instead of the Document Type Definition (DTD). Using XSD, we define the legal building blocks of the XML-based FAR document.

4. Implementation

4.1 Outline of the project

FAR is the folder for our project. It is located in the same directory as J2SE (Java 2 Standard Edition) which sets up the Java environment. It includes all the items described in the reminder of this section.

FAR is the main Java program to generate the user interface for the faculty members to edit their reports by filling out various forms. It uses JTree to create the tree-like interface.

HtmlFiles folder stores all the requirements listed in the Faculty Activity Report guidelines in plain text. When a leaf (subcategory) is selected, the corresponding requirement will appear in the HTML display area, *instruJSP* (described in 4.2) in the ***FAR***. The users can also follow the steps to create their reports in the Editor area. Also the user can get help text to complete the form if more information is needed.

Forms folder has all the .class files for each of the fifty-six forms for the project. The fifty-six forms depend on the categories of the guidelines of the Faculty Activity Report. When a leaf is selected, and the leaf stores a blank form, the blank form will pop up to allow the user to simply input information without creating the reports from scratch.

FilledForms folder is used to store the information that the user inputs and shows forms with information that has already input in the field.

Images folder has all the images for the functions in the toolbar such as save, open and close.

FAR.xsd is an XSD file that will be used to test the validity of XML file.

ConvertXMLToTree and ***ConvertXMLToTree*** are two files that allow users to convert any well-formed xml files to trees.

Xerces.jar is downloaded from <u>*xml.apache.org*</u>. It is used to support JDOM to parse an XML file.

4.2 Program components in the project.

In this section, we briefly explain the major components of the project[1] as follows:

(1) We set the native look and feel for the platform on which the program executes. The examples shown in this paper use the Windows Look And Feel.

(2) We set the content, FAR, in the Title Bar of the JFrame *far* (which is the outermost frame of the project) and *far*'s properties such as size and visibility.

(3) We create the JMenuBar, JMenu, and JMenuItem. We use the JMenu *File* and JMenuItem *New* as an example.

(4) We create the JToolBar *toolbar* with method *createToolbar* and add *toolbar* to the JFrame *far*. Also we create JButton *newButton* as an example and add it to the JToolBar *bar*.

(5) We create the JSplitPane *farPanel* (which is the outermost JSplitPane in JFrame *far*), JTextPane *info*, JEditorPane *instruJSP*, and JScrollPane *infoFP*

(6) We initialize the JTree *tree* and JScrollPane *treeJSP* that is to hold *tree*.

(7) We create JTree *tree* by adding DefaultMutableTreeNode to it.

(8) We set selectionMode and selectionListener.

(9) We create treeModel and set treeModelListener.

(10) We display the instructions for a selected category in HTML format in *instruJSP*.

(11) We display forms for a selected category.

(12) We create object *FARInfo*, which is used to specify the information of each tree node in JTree *tree*.

(13) We convert a well-formed XML document to JTree. We use two classes to do this: *ConvertXMLToTree.java* and *ConvertXMLToTreeTester.java*.

(14) We try to convert the JTree tree to XML document.

5. Conclusion

In our project, the technologies of Java Swing, XML, XSD, and JDOM are implemented to create an editing tool:

a) Swing provides all kinds of components that can be used to construct a sophisticated graphical user interface for a Java program. Also it implements the concept of the MVC model and improves it by combining the View and the Controller into a single UI object so that the tight coupling between View and Controller will be easy to be deal with. We used the Swing components to generate the user interface for the FAR Editor and all the needed forms.

b) XML is used to provide the file in a unique format so that queries can be readily performed on the document.

c) XSD is used to test whether the well-formed XML file is valid. One main advantage over DTD is that XSD is written in XML style and uses the XML syntax so that an XML editor, XML parser, and XML DOM can also be used by XML schemas. We used the ASBuilder dom sample program to test whether XML documents are valid.

d) JDOM is a tree-based, pure Java collection of APIs. It enables a program to parse a well-formed and valid XML document into a tree-like data structure and

[1] We changed the order of some sentences to give a clear explanation.

store it into memory in such a way that the XML data can be randomly accessed. We use it to retrieve the information from a well-formed and valid XML document to create a tree structure in memory. We then display the tree to the user so that the user can see the hierarchy of the XML data clearly. It is helpful for us to allow the user to work on the tree directly in our future work.

6. Future work

In the future, the following modifications to the FAR Editor should be carried out:

- Whenever a user clicks on any leaf of the *FilledForms* with *FilledInfo* stored in that, a Form with the stored information will appear allowing the user to modify it.
- The editing tool will support the incremental document creation.
- We can also use the XSLT for output. This would make fuller use of the MVC pattern that allows separating the view and application.

References:

[1] Sun Microsystems, About the JFC and Swing, Sun Microsystems, 2002, http://java.sun.com/docs/books/tutorial/uiswing/start/swingIntro.html.

[2] World Wide Web Cousortium, eXtensible Markup Language (XML) 1.0 (Second Edition), W3.org, 2000, http://www.w3.org/TR/REC-xml#dt-element

[3] World Wide Web Cousortium, Introduction to XML Schema, W3Schools.com, 2002, http://www.w3schools.com/schema/schema_intro.asp.

[4] JDOM.org, What is JDOM? JDOM.org, 2002, http://jdom.org/docs/faq.html

[5] Sun Microsystems, Advanced Programming for the Java^{TM} 2 platform, Sun Microsystems, 2002, http://developer.java.sun.com/developer/onlineTraining/Programming/JDCBook/swing.html

[6] And, M., Introducing Swing Architecture, Sun Microsystem, 2002, http://java.sun.com/products/jfc/tsc/articlesnews/getting_started/getting_started2.html

[7] Apache, Xerces Parser Readme, Apache, 2002, http://xml.apache.org/xerces2-j/index.html

[8] Bray, T., Paoli, J., Sperberg-McQueen, C., eXtensible Markup Language (XML) 1.0., W3.org, 2000, http://www.w3.org/TR/REC-xml.

[9] Deitel, H., & Deitel, P., *Java: How to Program*, New Jersey: Prentice Hall, 1999.

[10] Harold, E. R., Wrapping Your Own Packages, 2002, http://www.ibiblio.org/javafaq/course/

[11] Harold, E., Processing XML with Java, 2002, http://cafeconleche.org/books/xmljava/

[12] Sun Microsystems, Tech Tips, Sun Microsystems, 2001, http://developer.java.sun.com/developer/JDCTechTips/2001/tt0130.html.

[13] Walrath, K., Campione, M., *The JFC Swing Tutorial: A Guide to Constructing GUIs*. Boston: Addison-Wesley, July 1999.

[14] W3.org, 1997, Comparison of SGML and XML World Wide Web Consortium, http://www.w3.org/TR/NOTE-sgml-xml-971215.

[15] World Wide Web Cousortium, Introduction to XML, W3Schools.com, 2002, http://www.w3schools.com/xml/xml_whatis.asp.