Java prototype of hypercard bibliography past implementation and present choices

Neeta Reddy

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Computer Engineering Commons

Recommended Citation

Reddy, Neeta, "Java prototype of hypercard bibliography past implementation and present choices" (2002). Theses Digitization Project. 2071.
https://scholarworks.lib.csusb.edu/etd-project/2071

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
JAVA PROTOTYPE OF HYPERCARD BIBLIOGRAPHY - PAST IMPLEMENTATION AND PRESENT CHOICES

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Neeta Reddy
December 2002
JAVA PROTOTYPE OF HYPERCARD BIBLIOGRAPHY – PAST IMPLEMENTATION AND PRESENT CHOICES

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Neeta Reddy
December 2002

Approved by:

Richard Botting, Chair, Computer Science
George Georgiou
Tong/Lai Yu

Date: Nov/13/02
ABSTRACT

This project was started with JDK Version 1.0 and was later upgraded to version JDK Version 1.2.2, to create a graphical user interface using the Abstract Window Toolkit (AWT) to a HyperCard bibliography of software engineering. The bibliographic index tool is designed to facilitate searching for text and is run as a java applet. It presents an alphabetically ordered list of author names and Subjects. With the bibliography index tool one can manipulate a bibliographic list directly over the World Wide Web on a computer that lists electronic bibliographies. Timing has always been a critical factor in the world of Information technology and Java has made huge strides in this field with the advent of Swing Technology for GUI development. This project presents the past implementation and current choices that are available to users particularly pertaining to Java in order to allow them to come up with a better and newer system.
ACKNOWLEDGMENTS

I would like to thank Dr. George Georgiou and Dr. Tong Yu for serving on my committee and their insightful comments on the draft of this project. I am particularly grateful to Dr. Richard Botting, my project advisor, for his patience, advice, guidance, detailed suggestions and corrections of the design, implementation, and report of this project.

I would also like to thank my company ESRI (Environmental Systems Research Inc.) for allowing me to take some time off to complete this project.

The support of the National Science Foundation under award 981078 is gratefully acknowledged.

Last but not the least, I would like to thank my family and in particular my friends Raj Padmanabhan, Josefina Santiago, Payal Kamdar and Ashish Shah for their constant support for my continuing education and for this project in particular. My friends also spent many hours playing with our children so that I could work on this project.
DEDICATION

I would like to dedicate this project to my parents
Vasu and Shekar for encouraging me to complete it.
TABLE OF CONTENTS

ABSTRACT .................................................. iii
ACKNOWLEDGMENTS ........................................ iv
LIST OF FIGURES ........................................... vii

CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION

Introduction ..............................................  1
Purpose of the Project .................................  1
Context of the Problem ................................  2
Significance of the Project ............................  3
Limitations of the Project .............................  7
Definition of Terms ......................................  7
Organization of the Project .........................  8

CHAPTER TWO: SOFTWARE DESIGN

Development Environment .............................  9
Operating Environments ...............................  9
Maintenance Environments ........................... 10
Preliminary Functional Description ...............  10
Find Operation .......................................... 11
Select Operation ....................................... 11
Exit Operation ......................................... 12
Query Operation ........................................ 12
Life Cycle of an Applet .............................. 12

CHAPTER THREE: METHODOLOGY

Introduction .............................................. 14
User Interface Description ......................... 14
Main Window ........................................ 14
Author Display List Menu ............................ 15
The Subjects Display List Menu ...................... 17

CHAPTER FOUR: CURRENT OPTIONS
   Introduction ....................................... 26
   Alternative Technologies .......................... 26
   Using Java Server Pages or Servlets .......... 26
   Swing Graphical User Interface Client ........ 28
   Using Extensible Markup Language ............... 29
   Using Wireless Devices ............................ 30
   The Wikiwikiweb Source ............................ 31

CHAPTER FIVE: CONCLUSION .......................... 32

APPENDIX: CLASS DESCRIPTIONS ........................ 34

REFERENCES .......................................... 60
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Introduction Menu</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Author Display List Menu</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>Author Display Detail Menu</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Help Dialog</td>
<td>17</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>Subjects Display List Menu</td>
<td>18</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Subject Display Details Menu</td>
<td>19</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>AND Dialog</td>
<td>20</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>AND Results</td>
<td>21</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>OR Dialog</td>
<td>22</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>OR Results</td>
<td>23</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>NOT Dialog</td>
<td>24</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>NOT Results</td>
<td>25</td>
</tr>
</tbody>
</table>
CHAPTER ONE
SOFTWARE REQUIREMENTS SPECIFICATION

Introduction

The contents of Chapter One present an overview of the project. The purpose of the project is discussed followed by the context of the problem, and significance of the project. Finally, the limitations that apply to the project are reviewed followed by a brief description of how the project is organized.

Purpose of the Project

The purpose of this project is to provide a faster access to a HyperCard [9] bibliography of software engineering literature. Dr. Richard Botting of California State University, San Bernardino developed the original prototype of the bibliographic system. He used a Macintosh and HyperCard, which is Apple Computer Inc.'s application building tool. Using HyperCard one can build applications that incorporate the features of standard Macintosh applications - Pull down menus, multiple windows, pop up dialogs, buttons and multifont text boxes to produce distinctive applications with their own look and feel. Such "hyper-programming" systems are good for developing
programs, which users interact directly through a GUI. But, hyper-programming does not suit all programming requirements as the scope of organizing objects is quite limited and the system has correspondingly more work to do. This need for a more versatile language lead to the selection of Java as the ideal programming language.

The main purpose of selecting Java over the other existing languages is because it is extremely simple, object oriented, and integrated with the Internet. The neutral architecture and portable aspects of the Java language make it an ideal development language to meet the challenges of distributing dynamically extensible software application across networks.

Context of the Problem

In Dr. Botting’s Macintosh prototype any set of subjects can be selected and lists of items matching any number of these topics are easily generated. This list can then be filtered by key word matches or with other topics. This has proved itself capable of generating bibliographies for people asking for all kinds of topics on the Usenet newsgroup comp.software-eng. Given a list of items it is also possible to display (and select) the topics or subjects common to all of them. This is all
done with an intuitive GUI using HyperCard. It is however very slow.

The database has been uploaded as two linked HTML files to the department’s web server. They can be browsed but take a long time to load. A simple searching mechanism can be used to find items matching keywords. However, these lists of retrieved information are on a page and so resist further manipulation by the user. Using Java to download, display and refine a list of items should make this system as useful as the Macintosh prototype.

A recently published article shows that WWW search engines are less than ideal tools for searching [4]. Firstly, key word searching doesn’t work well - too many “false drops” are generated - hundreds of thousands items are typical. Each search is treated as a separate transaction and cannot be refined. Finally there is no simple way to track what has been found and how different strategies compare [1].

Significance of the Project

The World Wide Web is a suitable platform for a globally available database but proves to be slow for transferring large quantities of data. Downloading the
whole bibliography database which now contains 3000 plus items for a local search takes too long. Instead, the server will extract relevant items and let the client download them. Java will be used to take advantage of client server technology on the Internet to allow the user to select and view parts of this bibliographic database. This is a prototype that will be evaluated before final implementation because user satisfaction is the critical factor here. The main purpose is to satisfy the user and to come up with a bibliographic system that is -

1. Faster,
2. Easy to use by a novice user, and
3. Powerful

It is intended to provide a more strongly focused search facility than that provided by typical WWW Search Engines. A user will then be able to combine and refine previous searches.

Here is a situation that this scenario would be in. A user recalls that a particular result is in an old (1950’s) book on computer theory by an author called Davies. She needs to find the actual book and the full name of the author. A search using the CSUSB OPAC online catalog shows that there are over 500 books that are authored by “Davies” and it is difficult to select from
this list those books, on the theory of computation. A search for theory discovers hundreds of papers and once again it is difficult to select from this, the books by Davies. Searches for more detailed theoretical titles find two or three books but none by a Davies. The MELVYL system at CSUSB allows one to search several indexes and take the intersection. For example, the cryptic command -FIND PA DAVIES AND TW THEORY will first find all Personal Authors "Davies" and then all books with a Title Word, and then collate the two lists. However the word "THEORY" has too many entries in the index of title words for the search to complete in a useful time. However, FIND PA DAVIES AND TW THEORY COMPUT* does work and uncovers five books by MARTIN DAVIES all published in the late 70's and early 80's. Returning to CSUSB OPAC and searching "A=Martin Davies" find nothing, but "A=Davies Martin" produces the book that the user wanted.

In a better system it would be possible to input DAVIES into a window and click a button to get a list of items by that author. Then entering THEORY and selecting another button could discard items that do not have this title. These could be sorted by date (another button) and the user could edit the list down to the likely items and then request the details. Finally, the user can use copy
and paste to keep the results of his library searched for future work.

In Dr. Botting's prototype system all items by Davies are listed. This list can be selected and then the entries for each item can be searched for specific strings. Only items meeting these criteria remain. The aim of this project is to provide a similar functionality - to allow people to manipulate the results of searches directly.

Here is a second scenario where the proposed system would be helpful. Suppose a graduate student is searching the literature for papers on regular expressions and Turing Machines. There is a large database of computer science papers with a search engine on the web. It is easy to retrieve hundreds of papers on either topic. But it is not possible to take a pair of bibliography databases and combine them.

Dr. Botting hoped that the ideas he prototyped on the Macintosh could be ported to a Java system that uses a Common Gateway Interface to access a bibliography database. This should be faster to search than the current system and also provide better searching capabilities than a normal search engine.

In summary, the new system is needed whenever the searcher has several kinds of clues about a body of work
and needs to bring all of these to bear on the search. The idea is to give the user the power to collect, refine, edit, compose, and in general, manipulate the bibliographic lists directly over the World wide Web on a computer that lists electronic bibliographies.

Limitations of the Project

The project was originally started in 1997 and used the AWT components of JDK Version 1.0. The development environment was later upgraded to use JDK version 1.2.2. The version of Java and the use of AWT in GUI development would be termed a limitation in this current day and age in the sense that the latest version of Java makes use of Swing that offers forms based look and feel that is more efficient and has more capabilities. Swing components have a cleaner look and feel than the Abstract Window Toolkit (AWT) that was used in this project [8].

Definition of Terms

The following terms are defined as they apply to the project.

GUI     Graphical user Interface
ESRI    Environmental Systems Research Institute.
Java    One of the most popular software development languages.
Jarred
Result of creating a Java Archive

XML
Extensible Markup Language, a meta-markup language for text documents.

CGI
Common Gateway Interface

WWW
World Wide Web

HTML
Hypertext Markup Language

JSP
Java Server Pages

WAP
Wireless Application Protocol

AWT
Abstract Window Toolkit

Organization of the Project

The project was divided into five chapters. Chapter One provides software requirement specifications, an introduction to the context of the problem, purpose of the project, significance of the project and limitations. Chapter Two consists of the software design. Chapter Three documents the detailed methodology of the project. Chapter Four presents the alternative technologies that can be researched in the context of the project. Chapter Five presents the conclusions followed by the Appendix describing the class descriptions. This is followed by the references.
CHAPTER TWO
SOFTWARE DESIGN

Development Environment

User input modules will be kept separate from database modules. Development of this product will make use of the following software and hardware -

1. Java Development Kit 1.0 (Used for original development)
2. Sun Sparc 10 (Used for original development)
3. Special Servers - ftp = WWW, Web
4. IBM Based Compatible PC (Used for testing - original development)
5. Windows 2000 development environment
6. 100 MHz 486/DX4 (For original documentation and testing)
7. Standard Vi editor and Unix tools.
8. Netscape Communicator Version 4.74
9. Visual Café 1.4 (used to upgrade the code)
10. Java SDK 1.2.2

Operating Environments

Operation of this product shall be performed within the environment as specified above in the section on Development Environment.
Maintenance Environments

Maintenance of this product shall be performed within the complete environment as specified in the section on Development Environment.

Preliminary Functional Description

The system is designed to perform the following different types of operations viz., that of Find, Select, Exit, AND, OR, NOT. The GUI has been designed using Java’s Abstract Window toolkit (AWT) version 1.4. In Java, the various GUI building blocks are usually called components. The idea is that one should create the user interface with the various building blocks such as buttons, input areas for text (text fields) and scroll bars. The interface can then be programmed to respond to the various events. Components, in turn, are placed inside a java object called a container. The application is then run as an applet within a web page via a java-enabled browser such as Netscape 2.0 or above or Internet Explorer 3.0 or above. In order to load the applet into the web browser one must create a separate file that contains HTML tags to tell the browser which applets to load and where to put each applet on the Web page.
The java extensions to HTML tell any java-enabled browser the following -

1. The name of the class file.
2. The location of the class file.
3. How the applet sits on the web page.

The browser then retrieves the class file from the Net or the directory on the user’s machine and automatically runs the applet.

Find Operation

This operation returns a set of bibliographic items in a specific order.

Select Operation

A user can view the bibliographic information. The HyperCard bibliography basically contains items, words and topics. There is a many to many relationship in the occurrence of items to words. Words can occur in many items and vice versa. An item has approximately 30 words. Topics are specialized words and currently there are approximately 5 topics per word with about 1000 relevance links. This is how the system works.

1. The user selects a word from a scrollable list.
2. The system will generate a list of items containing the occurrences of the word.
3. The user can also typically select an item from a scrollable list.

4. The system will then generate a list of words or topics that has occurred under that item.

5. The user might then want to jump from a topic to its relevant occurrences.

6. The system will then display a list of relevant topics and so on.

Exit Operation

This operation allows the user to exit from the system.

Query Operation

The user can query the bibliography based on a number of Boolean operations like the AND, NOT and OR operation.

Life Cycle of an Applet

There are four methods in an applet class that gives it its framework. The following is a brief description of the four different stages of an applet.

1. init - This method is used for whatever initializations are needed for the applet.

2. Start - This method is automatically called after java calls the init method. This method is
where one usually restarts a thread for the applet.

3. Stop - This method is automatically called when the user exits off of the page on which the applet resides.

4. Destroy - Java calls this method when the browser shuts down normally. Since applets are meant to live on an HTML page, one need not worry about destroying the panel.
CHAPTER THREE

METHODOLOGY

Introduction

Included in Chapter three is a presentation of the methodology including the User Interface Description of the project.

User Interface Description

Main Window

The main screen (Figure 1) of the bibliographic search engine gives a brief introduction about the system.

The bibliography index tool is designed to facilitate searching for text. This demo uses the hypercard bibliography of the software engineering database being maintained by Dr. Richard Botting at CSUSB. The bibliography index tool presents an alphabetically ordered list of author names and subjects which can be manipulated directly over the world wide web.

Figure 1. Introduction Menu
The user has the option to either select "Start" to start
the demo or exit from the system at anytime by selecting
the "EXIT" button.

When the user selects the "START" button the
following screen comes up (Figure 2).

**Author Display List Menu**

This menu (Figure 2) has the following buttons -
"Find", "Subjects", "Help", "Exit". A list area is
displayed to the left of the screen that lists author
names that are available in the database.

![Figure 2. Author Display List Menu](image)

In order to access the bibliography, the user needs
to double click on one of the author names from the list
area. The selected item will then be displayed in the text field above. When the user selects the “Find” button the following screen comes up (Figure 3).

![Figure 3. Author Display Detail Menu](image)

The list item is a string that is URL encoded and the relevant item or items that is associated with that URL is then displayed in the text area on the right. The user can select the “Help” button either from the Author Display List Menu (Figure 2) or the Subject Display List menu (Figure 5). The “Help” button gives the user a brief description of all the buttons that are present in the
The following dialog (Figure 4) comes up when the user selects the "Help" button.

![Help Options]

- **Find**: Returns all records that contain the specified search string.
- **Subjects**: Displays the list of subjects.
- **AND**: Returns a list of authors that match the subject selected and the string entered in the AND dialog box.
- **NOT**: Returns a list of authors that match the subject selected but not the string entered in the NOT dialog box.
- **OR**: Returns a list of authors that match the subject selected or the string entered in the OR dialog box.

Figure 4. Help Dialog

The next important screen in the system is the Subjects Display List Menu. When the User selects the "Subjects" Button from the Display Author List Menu (Figure 2), the following menu comes up (Figure 5).

**The Subjects Display List Menu**

This menu has the following buttons - "Find", "And", "Not", "Or", "Previous", and "Help". A list area is displayed to the left of the screen that lists all the Subject names that are available in the database.
In order to access the Subjects bibliography, the user needs to double click on one of the subject names from the list area. The selected item will then be displayed in the text field above. When the user selects the "Find" button the following menu comes up (Figure 6).

The list item is a string that is URL encoded and the relevant item or items (author names) that is associated with that URL is then displayed in the text area on the right. The user then has the option to lookup the author names from the result by selecting the "PREVIOUS" button and navigating to the Author Display List Menu (Figure 2) from where he can start all over again.
Figure 6. Subject Display Details Menu

The user also has the choice to perform Boolean queries AND, NOT and OR on the list of Subjects. If he selects the “AND” Button the following dialog - “And What:” comes up prompting the user to enter the additional subject name.
When the user selects the "OK" button, the results of the Boolean query are displayed back in the Subjects menu Text area (Figure 8). The results also display the number of authors that contain the search string.
Figure 8. AND Results

When the user selects the “OR” Button the following dialog - “or What:” [Figure 9] comes up prompting the user to enter the additional subject name.
Figure 9. OR Dialog

When the user selects the "OK" button, the results of the Boolean query are displayed back in the Subjects menu Text area (Figure 10). The results also display the number of authors that contain the search string.
When the user selects the “Not” Button the following dialog - “And Not:" comes up prompting the user to enter the subject name that he does not want to include in his search.
Figure 11. NOT Dialog

When the user selects the "OK" button, the results of the Boolean query are displayed back in the Subjects menu Text area (Figure 12). The results also display the number of authors that contain the search string.
Figure 12. NOT Results
CHAPTER FOUR
CURRENT OPTIONS

Introduction

Included in Chapter four is a presentation of the alternative technologies that can be researched further in the context of this project.

Alternative Technologies

We will be examining the following technologies - HTML with JavaServer Pages and servlets, interactive Web via applets, using XML, using wireless devices, and lastly using the Wikiwikiweb [6] resource.

Using Java Server Pages or Servlets

This project was implemented using CGI scripts as a backend. A CGI database script is an external program run by the web server to access a database and create output for presentation on a web client. CGI is a proven architecture, but it has had four major drawbacks that create significant problems:

1. Difficulty in maintaining state and session Connections
2. Performance bottlenecks.
3. Can involve proprietary APIs
4. Security issues with scripts handling tainted data

There is a newer architecture that not only solves these problems, but also gives code portability, plus the ability to allow your server-side application to interface with a wide range of databases. This architecture uses JDBC with Java servlets to replace CGI. JDBC is a Java-based API. Servlets are server-side Java components that can access a database and output HTML or interface with JSP (Java Server Pages). They are protocol- and platform-independent components. Servlets save state information, can use a standard database API (i.e. JDBC), and have a significant performance increase because they have no heavy process startup and initialization for each client request like CGI does.

The advantages of using Java to develop an HTML front end is as follows - It is a great choice for relatively static content, if the interface needs to work for all user types running different software, if users access the application from slow networks and if one wants to quickly build an unsophisticated system.
Swing Graphical User Interface Client

Another emerging way of deploying Java applications is by using Sun's Java Web Start. It is similar in nature to Java Plug-in, but differs mostly in the first step. Java Web Start requires manual installation on each desktop machine, which is more tedious than the browser's automatic installation of the plug-in. Java's Web Start comes with the JRE and is rather easy to set up. Once it is set up, the applications that rely on Web Start can be downloaded and installed. Just as with a plug-in, the application is jarred and published on the Web. An HTML page lets the user launch the downloaded application to the user machine if it is not yet available. Downloaded applications are cached and can be launched independently through Java Web Start Application Manager, which looks very much like the Program Manager of Windows 3.1. Java Plug-in is simpler to set up and is user-friendlier than Java Web Start because it requires much less involvement from administrators and users alike. Summing up, with Java Plug-in and Java Web Start, the deployment of Swing applets is tremendously easier and safer than what it used to be, but is still more involved than just clicking on an HTML page with a bit of JavaScript. Some users may feel intimidated by going through the steps required to install
the JVM on their local machines and may not get to see Swing's benefits. But if one needs a dynamic GUI interface that gives a lot of flexibility to users, there is no better way than using a Swing applet [9].

Using Extensible Markup Language

Another area that can be investigated in the light of this project is to see if we can map the bibliography into an XML formatted document and provide search and submit interfaces.

XML (Extensible Markup Language) is a set of syntax rules and guidelines for defining text-based markup languages. XML languages have a number of uses including:

1. Exchanging information
2. Defining document types
3. Specifying messages

Information that is expressed in a structured, text-based format can easily be transmitted between, transformed, and interpreted by entities that understand the structure. In this way XML brings the same cross-platform benefits to information exchange as the Java programming language has for processing. Java Server Pages (JSP) technology provides specification and serving of documents that combine static markup language elements and
elements created dynamically by Java programming language objects. JSP pages executed as Java servlets, typically in the execution context of a web server where they generate content-dependent response documents using data stored in databases and in other server-based application objects.

JSP technology provides a number of capabilities that are ideally suited for working with XML. JSP pages can contain any type of text-based data, so it is straightforward to generate documents that contain XML markup. Also, JSP pages can use the full power of the Java platform to access programming language objects to parse and transform XML messages and documents. In particular, as part of the Java software environment, JSP pages can use objects that leverage the new Java APIs for processing XML data. Finally JSP technology provides an abstraction mechanism that encapsulates functionality for ease of use within a JSP page [5].

Using Wireless Devices

Wireless devices such as Palm Pilots, Pocket PCs and Wireless Application Protocol (WAP) enabled cell phones make it practical to provide anywhere, anytime access to applications that previously have required terminals or personal computers. The critical benefit for all of these
applications is they reduce the reaction time of mobile professionals who are far from their desks by providing real-time information to people and giving them a tool to take immediate action. Researchers often need to access bibliographies at home or in a seminar. Several recent projects at CSUSB have shown the feasibility of wireless access to the Web. The Boolean query technology of this project may be highly applicable in this wireless world [5].

The Wikiwikiweb Source

We can also use the WikiWikiWeb [6] interface to share bibliographies. Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has simple text syntax for creating new pages and cross-links between internal pages on the fly.

Wiki is unusual among group communication mechanisms in that it allows the organization of contributions to be edited in addition to the content itself [6].

Wiki supports the concepts of "open editing" which encourages the democratic use of the web. No work has been done on Boolean queries within a Wikiwikiweb system. This option would need to be researched.
CHAPTER FIVE
CONCLUSION

Early Web pages were able to offer interactivity only by running CGI scripts on the server. The advent of Java in 1995 provided a way to create scripts on the client side with certain intelligence. Since then, a great deal of effort has been expended to allow Java client-side programs to communicate with gateways and backend databases.

This project is a prototype of Java-CGI hybrid application - a java applet that can run selected CGI routines to interface with the server services. There are a lot more options present now than when the project was originally started. One can choose between Swing based, HTML with JSF/Servlets or even XML-based front ends for their applications. Another good thing that happened over the years to browser-based java is Sun's Java plug-in.

Each front-end technology has its benefits and disadvantages. For each application, one must perform an evaluation of these technologies with analysis and user expectations. The implementation of the HyperCard bibliography was done during the early days of Java where one had to fight with the differences among browsers and
applet download time that marred the picture. Things have changed tremendously since then.

Sun Microsystems has spent tremendous time improving its Java code. An attempt has been made of discussing the past implementation and the current choices that the user is faced with in order to come up with a better system that is more in-line with the current trends.
APPENDIX

CLASS DESCRIPTIONS
CLASS DESCRIPTIONS

addNotify() - Method in class JApplet1.AndDialog

addNotify() - Method in class JApplet1.OrDialog

addNotify() - Method in class JApplet1.NotDialog

addNotify() - Method in class HelpDialog

AndButton - Variable in class JApplet1

AndButton MouseClicked(MouseEvent) - Method in class JApplet1

AndDialog WindowClosing(WindowEvent) - Method in class JApplet1.AndDialog

AndDialog WindowClosing(WindowEvent) - Method in class JApplet1.OrDialog

AndDialog WindowClosing(WindowEvent) - Method in class JApplet1.NotDialog

AndLabel - Variable in class JApplet1.AndDialog

AndLabel - Variable in class JApplet1.OrDialog

AndLabel - Variable in class JApplet1.NotDialog

AndOk - Variable in class JApplet1.AndDialog

AndOk - Variable in class JApplet1.OrDialog
AndOk - Variable in class JApplet1.NotDialog

AndOk MouseClicked(MouseEvent) - Method in class JApplet1.AndDialog

AndOk MouseClicked(MouseEvent) - Method in class JApplet1.OrDialog

AndOk MouseClicked(MouseEvent) - Method in class JApplet1.NotDialog

AndText - Variable in class JApplet1.AndDialog

AndText - Variable in class JApplet1.OrDialog

AndText - Variable in class JApplet1.NotDialog

AuthorPanel - Variable in class JApplet1

AuthorPanel ComponentShown Interaction1(ComponentEvent) - Method in class JApplet1

AuthorPanel ComponentShown(ComponentEvent) - Method in class JApplet1

C

componentShown(ComponentEvent) - Method in class JApplet1.SymComponent

D

DetailText - Variable in class JApplet1
**ExitButton** - Variable in class JApplet

**ExitButton MouseClicked(MouseEvent)** - Method in class JApplet

Invoked when the mouse has been clicked on the Exit Button

---

**fComponentsAdjusted** - Variable in class JApplet1.AndDialog

**fComponentsAdjusted** - Variable in class JApplet1.OrDialog

**fComponentsAdjusted** - Variable in class JApplet1.NotDialog

**fComponentsAdjusted** - Variable in class HelpDialog

**FindButton** - Variable in class JApplet

**FindButton MouseClicked(MouseEvent)** - Method in class JApplet

This method takes the contents of a URI that does a Find operation on the Author list, parses it and displays the Detailed results in a Text area

**FindSubject** - Variable in class JApplet

**FindSubject MouseClicked(MouseEvent)** - Method in class JApplet

Will take contents of window as a URL attempt to parse it, and connect to the resource located by the URL

---

**HelpButton** - Variable in class JApplet

**HelpButton MouseClicked Interaction1(MouseEvent)** - Method in class JApplet
HelpButton MouseClicked(MouseEvent) - Method in class JApplet1

HelpDialog - class HelpDialog.

HelpDialog WindowClosing(WindowEvent) - Method in class HelpDialog

HelpDialog.SymMouse - class HelpDialog.SymMouse.

HelpDialog.SymMouse(HelpDialog) - Constructor for class HelpDialog.SymMouse

HelpDialog.SymWindow - class HelpDialog.SymWindow.

HelpDialog.SymWindow(HelpDialog) - Constructor for class HelpDialog.SymWindow

HelpDialog(Frame) - Constructor for class HelpDialog

HelpDialog(Frame, boolean) - Constructor for class HelpDialog

HelpDialog(Frame, String, boolean) - Constructor for class HelpDialog

init() - Method in class JApplet1

Instantiates and initializes the GUI components, register listeners and declare controls.

ItemNameField - Variable in class JApplet1

JApplet1 - class JApplet1.

JApplet1.java - a class for demonstrating the use of the Bibliography Index tool.
JApplet1.componentShown(ComponentEvent) - Method in class JApplet1

This method takes the contents of a URI, parses it and displays the author names in a List area.

JApplet1.AndDialog - class JApplet1.AndDialog.

This class represents a Dialog to get user response to the Boolean query AND.

JApplet1.AndDialog.SymMouse - class JApplet1.AndDialog.SymMouse.

JApplet1.AndDialog.SymMouse(JApplet1.AndDialog) - Constructor for class JApplet1.AndDialog.SymMouse.

JApplet1.AndDialog.SymWindow - class JApplet1.AndDialog.SymWindow.

JApplet1.AndDialog.SymWindow(JApplet1.AndDialog) - Constructor for class JApplet1.AndDialog.SymWindow.

JApplet1.AndDialog(JApplet1, Frame) - Constructor for class JApplet1.AndDialog.

JApplet1.AndDialog(JApplet1, Frame, boolean) - Constructor for class JApplet1.AndDialog.

JApplet1.NotDialog - class JApplet1.NotDialog.

This class represents a Dialog to get user response to the Boolean query NOT.


JApplet1.NotDialog(JApplet1, Frame) - Constructor for class JApplet1.NotDialog.

JApplet1.NotDialog(JApplet1, Frame, boolean) - Constructor for class JApplet1.NotDialog.
JApplet1.NotDialog(JApplet1, Frame, String, boolean) - Constructor for class JApplet1.NotDialog

JApplet1.OrDialog - class JApplet1.OrDialog.

This class represents a Dialog to get user response to the Boolean query OR


JApplet1.OrDialog.SymMouse(JApplet1, OrDialog) - Constructor for class JApplet1.OrDialog.SymMouse


JApplet1.OrDialog.SymWindow(JApplet1, OrDialog) - Constructor for class JApplet1.OrDialog.SymWindow

JApplet1.OrDialog(JApplet1, Frame) - Constructor for class JApplet1.OrDialog

JApplet1.OrDialog(JApplet1, Frame, boolean) - Constructor for class JApplet1.OrDialog

JApplet1.OrDialog(JApplet1, Frame, String, boolean) - Constructor for class JApplet1.OrDialog


A class representing the actions (hide/Display) performed on the AuthorPanel and SubjectPanel

JApplet1.SymComponent(JApplet1) - Constructor for class JApplet1.SymComponent


A class representing the actions performed on the various buttons on the window

JApplet1.SymMouse(JApplet1) - Constructor for class JApplet1.SymMouse

JApplet1() - Constructor for class JApplet1
**L**

- `label1` - Variable in class `JApplet1`
- `label2` - Variable in class `JApplet1`
- `label3` - Variable in class `JApplet1`
- `label4` - Variable in class `JApplet1`
- `label5` - Variable in class `JApplet1`
- `label6` - Variable in class `JApplet1`

**M**

- `MainExit` - Variable in class `JApplet1`
- `MainExitMouseClicked(MouseEvent)` - Method in class `JApplet1`
- `MainIntro` - Variable in class `JApplet1`
- `MainPanel` - Variable in class `JApplet1`
- `mouseClicked(MouseEvent)` - Method in class `JApplet1.SymMouse`
- `mouseClicked(MouseEvent)` - Method in class `JApplet1.AndDialog.SymMouse`
- `mouseClicked(MouseEvent)` - Method in class `JApplet1.OrDialog.SymMouse`
- `mouseClicked(MouseEvent)` - Method in class `JApplet1.NotDialog.SymMouse`
- `mouseClicked(MouseEvent)` - Method in class `HelpDialog.SymMouse`

**N**

42
NameList - Variable in class JApplet1

NameListMouseClicked(MouseEvent) - Method in class JApplet1

NotButton - Variable in class JApplet1

NotButtonMouseClicked(MouseEvent) - Method in class JApplet1

OkButton - Variable in class HelpDialog

OkButtonMouseClicked(MouseEvent) - Method in class HelpDialog

OrButton - Variable in class JApplet1

OrButtonMouseClicked(MouseEvent) - Method in class JApplet1

Previous - Variable in class JApplet1

PreviousMouseClickedInteraction1(MouseEvent) - Method in class JApplet1

This method takes the user to the previous menu

PreviousMouseClicked(MouseEvent) - Method in class JApplet1

setVisible(boolean) - Method in class JApplet1.AndDialog

Shows or hides the component depending on the boolean flag b.
setVisible(boolean) - Method in class JApplet1. OrDialog
Shows or hides the component depending on the boolean flag b.

setVisible(boolean) - Method in class JApplet1. NotDialog
Shows or hides the component depending on the boolean flag b.

setVisible(boolean) - Method in class HelpDialog
Shows or hides the component depending on the boolean flag b.

StartButton - Variable in class JApplet1

StartButton_MouseClicked() - Method in class JApplet1

SubjectButton - Variable in class JApplet1

SubjectButton_MouseClicked() - Method in class JApplet1

SubjectDetail - Variable in class JApplet1

SubjectHelp - Variable in class JApplet1

SubjectHelp_MouseClicked() - Method in class JApplet1

SubjectList - Variable in class JApplet1

SubjectList_MouseClicked() - Method in class JApplet1

SubjectPanel - Variable in class JApplet1

SubjectPanel_ComponentShown() - Method in class JApplet1

This method takes the contents of a URI, parses it and displays the Subject names in a List area
**SubjectTestField** - Variable in class JApplet1

**TextArea1** - Variable in class HelpDialog

**windowClosing(WindowEvent)** - Method in class JApplet1.AndDialog.SymWindow

**windowClosing(WindowEvent)** - Method in class JApplet1.OrDialog.SymWindow

**windowClosing(WindowEvent)** - Method in class JApplet1.NotDialog.SymWindow

**windowClosing(WindowEvent)** - Method in class HelpDialog.SymWindow
public class JApplet1
extends javax.swing.JApplet

JApplet1.java - a class for demonstrating the use of the Bibliography Index tool. It is a basic extension of the javax.swing.JApplet class

Version:
1.0

Author:
Neeta Reddy

See Also:
Serialized Form
### Inner Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JApplet1.AndDialog</td>
<td>This class represents a Dialog to get user response to the Boolean query AND</td>
</tr>
<tr>
<td>JApplet1.NotDialog</td>
<td>This class represents a Dialog to get user response to the Boolean query NOT</td>
</tr>
<tr>
<td>JApplet1.OrDialog</td>
<td>This class represents a Dialog to get user response to the Boolean query OR</td>
</tr>
<tr>
<td>JApplet1.SymComponent</td>
<td>A class representing the actions (hide/Display) performed on the AuthorPanel and SubjectPanel</td>
</tr>
<tr>
<td>JApplet1.SymMouse</td>
<td>A class representing the actions performed on the various buttons on the window</td>
</tr>
</tbody>
</table>

### Inner classes inherited from class javax.swing.JApplet

- javax.swing.JApplet.AccessibleJApplet

### Inner classes inherited from class java.awt.Component

- java.awt.Component.AWTTreeLock
<table>
<thead>
<tr>
<th>Fields inherited from class <code>javax.swing.JApplet</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>accessibleContext, rootPane, rootPaneCheckingEnabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from class <code>java.applet.Applet</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>serialVersionUID, stub</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from class <code>java.awt.Panel</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>base, nameCounter, serialVersionUID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from class <code>java.awt.Container</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>component, containerListener, containerSerializedDataVersion, dispatcher, layoutMgr, maxSize, ncomponents, serialVersionUID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from class <code>java.awt.Component</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>actionListenerK, adjustmentListenerK, appContext, assert, background, BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, changeSupport, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, containerListenerK, cursor, dropTarget, enabled, eventMask, focusListener, focusListenerK, font, foreground, hasFocus, height, incRate, inputMethodListener, inputMethodListenerK, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, minSize, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, name, nameExplicitlySet, newEventsOnly, ownedWindowK, parent, peer, peerFont, popups, prefSize, RIGHT_ALIGNMENT, serialVersionUID, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowListenerK, x, y</td>
</tr>
</tbody>
</table>
### Constructor Summary

**JApplet1()**

### Method Summary

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>AndButton_MouseClicked</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>AuthorPanel_ComponentShown_Interaction1</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>AuthorPanel_ComponentShown</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>ExitButton_MouseClicked</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>FindButton_MouseClicked</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>FindSubject_MouseClicked</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>HelpButton_MouseClicked_Interaction1</strong></td>
</tr>
<tr>
<td>(package private)</td>
<td>void</td>
<td><strong>HelpButton_MouseClicked</strong></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>init()</td>
<td></td>
</tr>
<tr>
<td>Instantiates and initializes the GUI components, register listeners and declare controls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>JApplet1_componentShown</td>
<td>This method takes the contents of a URI, parses it and displays the author names in a List area</td>
</tr>
<tr>
<td>(package private) void</td>
<td>MainExit_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>NameList_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>NotButton_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>OrButton_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>Previous_MouseClicked</td>
<td>Interaction1</td>
</tr>
<tr>
<td>This method takes the user to the previous menu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>Previous_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>StartButton_MouseClicked</td>
<td>Interaction1</td>
</tr>
<tr>
<td>This method takes the user to the Author Display list Panel.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>StartButton_MouseClicked</td>
<td></td>
</tr>
<tr>
<td>(package private) void</td>
<td>SubieqtButton_MouseClicked</td>
<td>Interaction3</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>void (java.awt.event.MouseEvent event)</code></td>
<td>This method takes the user to the Subjects Display list Panel.</td>
<td></td>
</tr>
<tr>
<td>(package private) void <code>SubjectButtonMouseClicked (java.awt.event.MouseEvent event)</code></td>
<td>Invoked when the mouse has been clicked on the Subject Button</td>
<td></td>
</tr>
<tr>
<td>(package private) void <code>SubjectHelpMouseClicked (java.awt.event.MouseEvent event)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(package private) void <code>SubjectListMouseClicked (java.awt.event.MouseEvent event)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(package private) void <code>SubjectPanel_ComponentShown (java.awt.event.ComponentEvent event)</code></td>
<td>This method takes the contents of a URI, parses it and displays the Subject names in a List area</td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class `javax.swing.JApplet`

- addImpl, createRootPane, createRootPaneException, getAccessibleContext, getContentPane, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent, remove, getContentPane, getGlassPane, getJMenuBar, getLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

### Methods inherited from class `java.applet.Applet`

- destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop
### Methods inherited from class `java.awt.Panel`

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>addNotify, constructComponentName</td>
</tr>
</tbody>
</table>

### Methods inherited from class `java.awt.Container`

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
</table>
setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, wait, wait, wait

**Field Detail**

**AuthorPanel**

def java.awt.Panel AuthorPanel

**ItemNameField**

def java.awt.TextField ItemNameField

**NameList**

def java.awt.List NameList

**DetailText**

def java.awtTextArea DetailText

**FindButton**

def java.awt.Button FindButton

**SubjectButton**

def java.awt.Button SubjectButton

**HelpButton**

53
java.awt.Button HelpButton

ExitButton

java.awt.Button ExitButton

label1

java.awt.Label label1

label2

java.awt.Label label2

label3

java.awt.Label label3

MainPanel

java.awt.Panel MainPanel

MainIntro

java.awt.TextArea MainIntro

StartButton

java.awt.Button StartButton

MainExit

java.awt.Button MainExit

SubjectPanel

java.awt.Panel SubjectPanel
AndButton

java.awt.Button AndButton

FindSubject

java.awt.Button FindSubject

NotButton

java.awt.Button NotButton

OrButton

java.awt.Button OrButton

Previous

java.awt.Button Previous

SubjectList

java.awt.List SubjectList

SubjectDetail

java.awt.TextArea SubjectDetail

SubjectTestField

java.awt.TextField SubjectTestField

SubjectHelp

java.awt.Button SubjectHelp

label4

java.awt.Label label4
Iabel5

java.awt.Label label5

Iabel6

java.awt.Label label6

**Constructor Detail**

JApplet1

public JApplet1()

**Method Detail**

init

public void init()
Instantiates and initializes the GUI components, register listeners and declare controls
Overrides:
init in class java.applet.Applet

FindSubject_MouseClicked

void FindSubject_MouseClicked(java.awt.event.MouseEvent event)
Will take contents of window as a URL attempt to parse it, and connect to the resource located by the URL

ExitButton_MouseClicked

void ExitButton_MouseClicked(java.awt.event.MouseEvent event)
Invoked when the mouse has been clicked on the Exit Button

SubjectButton_MouseClicked

void SubjectButton_MouseClicked(java.awt.event.MouseEvent event)
Invoked when the mouse has been clicked on the Subject Button
SubjectButton_MouseClicked_Interaction3

void SubjectButton_MouseClicked_Interaction3(java.awt.event.MouseEvent event)

AuthorPanel_ComponentShown

void AuthorPanel_ComponentShown(java.awt.event.ComponentEvent event)

AuthorPanel_ComponentShown_Interaction1

void AuthorPanel_ComponentShown_Interaction1(java.awt.event.ComponentEvent event)

JApplet1_componentShown

void JApplet1_componentShown(java.awt.event.ComponentEvent event)

This method takes the contents of a URI, parses it and displays the author names in a List area

FindButton_MouseClicked

void FindButton_MouseClicked(java.awt.event.MouseEvent event)

This method takes the contents of a URI that does a Find operation on the Author list, parses it and displays the Detailed results in a Text area

NameList_MouseClicked

void NameList_MouseClicked(java.awt.event.MouseEvent event)

HelpButton_MouseClicked

void HelpButton_MouseClicked(java.awt.event.MouseEvent event)

Previous_MouseClicked

void Previous_MouseClicked(java.awt.event.MouseEvent event)

Previous_MouseClicked_Interaction1

void Previous_MouseClicked_Interaction1(java.awt.event.MouseEvent event)
This method takes the user to the previous menu

SubjectPanel_ComponentShown

void SubjectPanel_ComponentShown(java.awt.event.ComponentEvent event)
This method takes the contents of a URI, parses it and displays the Subject names in a List area

SubjectList_MouseClicked

void SubjectList_MouseClicked(java.awt.event.MouseEvent event)

SubjectHelp_MouseClicked

void SubjectHelp_MouseClicked(java.awt.event.MouseEvent event)

AndButton_MouseClicked

void AndButton_MouseClicked(java.awt.event.MouseEvent event)

HelpButton_MouseClicked_interaction1

void HelpButton_MouseClicked_interaction1(java.awt.event.MouseEvent event)

OrButton_MouseClicked

void OrButton_MouseClicked(java.awt.event.MouseEvent event)

NotButton_MouseClicked

void NotButton_MouseClicked(java.awt.event.MouseEvent event)

MainExit_MouseClicked

void MainExit_MouseClicked(java.awt.event.MouseEvent event)

StartButton_MouseClicked

void StartButton_MouseClicked(java.awt.event.MouseEvent event)
StartButton_MouseClicked_Interaction1

void StartButton_MouseClicked_Interaction1(java.awt.event.MouseEvent event)
REFERENCES


   http://www.microsoft.com/usability/webconf/schlichting/schlichting.htm

[5] The source for white papers on Java and XML
   http://www.itpapers.com/

[6] The Wikiwikiweb site
   http://www/wikiwikiweb.com

[7] JDK 1.1 and Beyond
   http://developer.java.sun.com/developer/technicalArticles/GUI/Transition/

[8] What is Hypercard ?
   http://www.wired.com/news/school/0,1383,54365,00.html

[9] JAVA GUI’s
   http://developer.java.sun.com/developer/technicalArticles/GUI/frontend