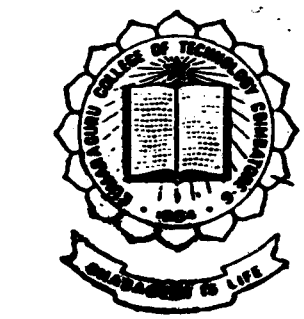
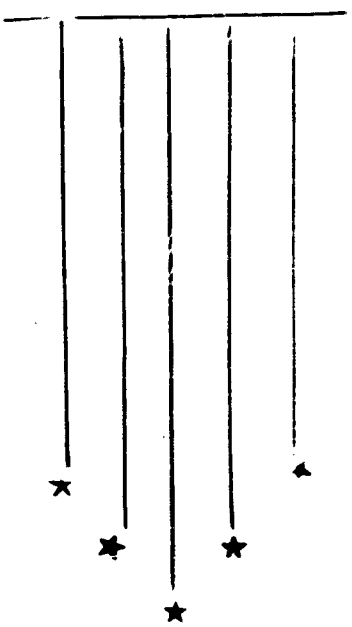
SEARCH ENGINE FOR AN INTRANET

Project Report





Submitted by

P. KRISHNAKUMAR

T. C. SATHEESH

K. VISAKAN

Under the Guidance of

Ms R. SUMATHI, B.E., M S.

In Partial fulfilment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the Bharathiyar University, Coimbatore

1997 - 98

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore-641 006

KUMARAGURU COLLEGE OF TECHNOLOGY

COMBATORE - 641 006.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to Certify that the Report entitled

"SEARCH ENGINE FOR AN INTRANET LIBRARY"

has been submitted by

Mr. P.KRISHNAKUMAR, T.C.SATHEESH & K.VISAKAN

in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in the Computer Science and Engineering Branch of the Bharathiar University, Coimbatore - 641 006 during the academic year 1997-'98

(Guide) 27/4/98

(Head of Department)

Certified that the Candidate was Examined by us in the Project Work Viva-Voce Examination held on 27-4-98 *and the University Register Number is* 9427K0133, 9427K0143, 9427K0153.

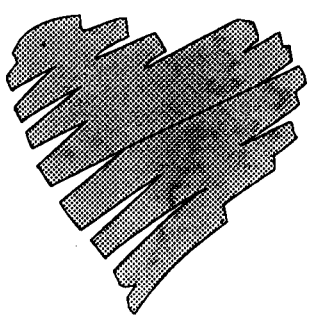
(Internal Examiner)

(External Examiner) 27/4/98

Dedicated to our

beloved

Parents



ACKNOWLEDGEMENTS

ACKNOWLEDGMENTS

Regarding this project we would like to acknowledge the help given to us by a number of people:

First of all we would like to thank the Principal **Dr. S. Subramanian** and the management for allowing us to use the College's Computer facilities for our Project.

We would like to thank our Head of Department Prof. **P. Shanmugam** B.E. M.Sc. (Engg), M.S (Hawaii) M.I.E.E.C, M.I.S.T.E for his help in providing us with the necessary software to do our project.

Our heartfelt gratitude to our guide Ms. **R.Sumathi B.E, M.S** who helped us in all the possible way from helping us to get the books, software in time and also for the constant tips and suggestions in improving the software.

We would also like to extend our sincere thanks to all the Staff members of the Computer Science & Engineering Department for all the help they have extended to us during the project.

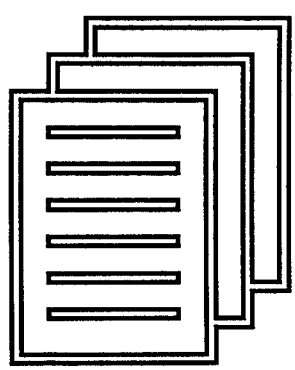
During this project we spent quite some time away from home or sulking at home brooding over our work. We would like to thank our parents for bearing our mood swings and providing constant encouragement so that we could successfully complete our project. We would also like to thank them for their help in this project. We would also like to thank our brothers and sisters for their help and encouragement.

Our special thanks to Ms. **Bagyalakshmi Ravindran,** Scientist Engineer, S.E. Computer Information Group, ISRO Banglore who gave us the idea for the project and hence sowed the seed for the present success.

We would also like to thank Ms. **Shanthi Mani**, Assistant Research Officer at Pasteur Institute of India Coonoor, for her help in this regard and also for her encouragement.

We would like to express a sincere gratitude to Ms. **Rajam Muralidharan,** Senior Librarian Information Assistant, Pasteur Institute of India, Coonoor, for the help she provided for us.

Last but in no way the least we would like to thank our friends who were a constant help and encouragement in this project and who share the success of it as much as we do.



CONTENTS

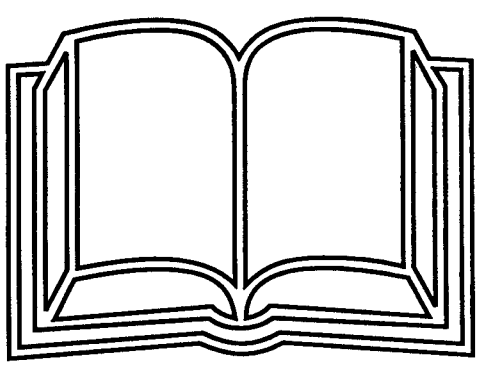
TABLE OF CONTENTS

SYNOPSIS

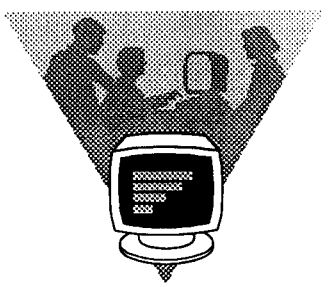
SYNOPSIS

The Demi! Search Engine is a software that enables a user in an organisation to search for html pages which are part of a library of reference materials. He can enter any topic so that he will be able to retrieve the material from the corresponding html page. Another option that the user has is to use the Search Page below the Search box where the general topics are listed with links that will enable the user to look and learn a new concept of refer from it.

The Search Engine has a database that has three tables that are used to store and link the topics and addresses. These are searched by the software and presented to the user.



INTRODUCTION



INTRANETS

INTRODUCTION

INTRANETS

Ever since the Internet has become a household name and a necessity for almost all organisations information has become the watchword in the industry. The vast amount of information that is available on the Net is not only available free of cost to the user but also can be utilised productively . But along this immense data came a problem of retrieval of the necessary data for the user or the organisation Searching for these data became lot easier when Search Engines were introduced. The user had to only type the topic he wishes for which he needed data and the Search Engine would give him a list of all the files that had the necessary data. This was helpful for the Internet.

When the concept and technology of the net were applied to the organisations network they were called Intranets. Intranets concentrated their effort on having unique Open system environment for the exchange of data and information with the organisation. Here again there arose a

necessity for having an efficient method to retrieve the data. Hence Search Engines became an essential here also.

The Intranet offers the technology of the Net with concepts of WWW, ftp, and email within the organisation without affecting the current network set-up. This helps for an open system in the exchange of regular data at the same time provides security for the classified pages.

The Intranet for an organisation has a Home Page that is the window to the information that is available to the employees of it. This Home Page is split into various categories that identify a particular part of the organisation. There are details available of the individual department activities and the notices that have been issued by the department to its employees.

To implement the Search Engine as part of the Home Page we have to include the Search window to the Home Page by adding the class file of the Java program and provide the necessary space for it in the Home Page can do this.

THE ELECTRONIC LIBRARY

Newspapers, books, magazines are all sources of information to the user. They are all necessary for adding and updating knowledge. But when working on a project it becomes tedious to keep the books in front of us if we ever get them and then work so that any time we do have a doubt we can refer to the book and clear the doubt. Another problem that occurs when using a book is that there is the problem of searching the data in the book . This has to be done manually and is very tedious and time consuming.

In order to make the work easier for the user we decided to create an electronic library that will help the user in his day to day work. There are two ways that we can accomplish this task:

1. We can direct the Search Engine topic addresses to the Internet and hence provide the library for the user.

2. We can create a library of our own in the server and provide the topic addresses to these pages of data.

There are a lot of advantages and disadvantages in these two methods.

The disadvantages of working with the Internet are:

1. When using the Internet for the library there arises the problem of the availability of the connection as and when the user requests it.

2. Another problem is the telephone bill that could keep accumulating due to the constant use of the phone for connection to the Internet.

3. Then there lies the problem of a barrage of the data that is available in the Net that could overwhelm any user. This could be advantageous sometimes but for a professional this could be a little too much.

The advantages are:

1. If the resources of the Internet are carefully studied and used then there can be no better library on earth. But this involves a very thorough study of the various data.

2. This eliminates the cost of setting up a personal library.

There are a lot advantages and disadvantages in these two methods.

The reasons why we would personally suggest a private library for each organisation are:

1. A survey among the user of this library could help in determining the necessary data that is to be maintained in the library. This is advantageous because this could help to keep the library size within manageable costs.

2. Then there is the advantage of the user having a regular and uninterrupted access to the library and also keeping the management tension free of the phone bills.

3. The library could be customised to the individual organisation's needs and necessities.

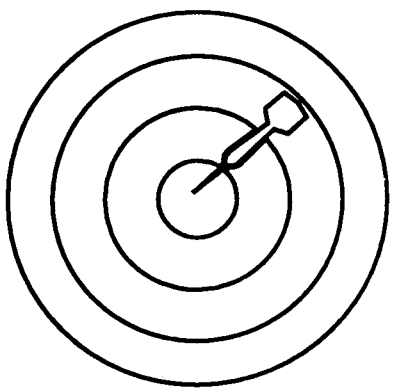
4. when creating the database for the topics in the library there could be a greater efficiency in the storage and retrieval of the data because of the size of the library. That is a user will not be bombarded by the data but instead will have a manageable amount of data pages.

The disadvantages that could arise from this process of creating an independent library are:

1. The size of the library becomes a restriction as this is directly proportional to the cost of the library and hence has to be kept at minimum tolerable levels.

2. Updating the library regularly is also a factor that has to be taken account of.

After considering these factors that affect the library we find that an independent library is a better option for the organisation.



THE PROJECT

THE DEMI! SEARCH ENGINE

The Demi! is a regular Search Engine which is designed to suit any organization's needs. This software has a very simple and easy to use environment that could be customized for the individual settings.

The Demi! has a Home Page that is used as the first-screen for the user and it acts as the gateway to the data of the Intranet. The Demi! has a text box , here the user can type his topic name and request a Search by clicking on the button that is next to the text box.

The Demi! searches its database and checks for the topic entry. If it is found then it locates the address for the topic and provides the result. If the result has numerous addresses then a separate screen will produce the results in the form of links where the topic can be assessed and a particular page can be called.

Besides this there is the Home Page itself which has a set of links that are arranged in alphabetical order and which provide a list of the general

topics that are available in the library. There is also the other Sub Home Pages of each department and other current news of the organisation.

The Demi! Search Engine was implemented using Microsoft Visual J++1.1. It is connected to a Project database that was created with Microsoft Access. The Search Engine window is checked for the topic name and compared with the topic table of the database. If the topic exists in the table then the heading of the related pages are put into the list box . When the user clicks on the list box on a particular topic he is able to retrieve the page from the library and read it. Each page has a particular set of hyperlinks that will help the user surf through the Intranet. There are also a set of See Also's where the user can surf through related topics.

The Home page can be incorporated to the Internet Explorer when installing the software hence every time the user opens the Internet Explorer he gets the Home Page first.

ABOUT DEMI! DATABASE

The software is made of database that was created using MS Access.

The database has a total of three tables.

1. Topic table.
2. Address table.
3. Topic & Address table.

The database has been split into three tables so as to remove data redundancy in the table. Since the topics and addresses can occur numerous times there arises a problem of the same address being used numerous times. This can create a problem so the connection is done using unique identifiers for each record in the Topic and Address table.

The Topic table:

This table has two fields:

1. Topic ID
2. Topic.

It is meant to keep the data about the topics that are to be searched and the number that relates it to the addresses.

The second table has a total of three fields:

1. Address ID
2. Address
3. Page Header.

They hold the addresses of the pages of the library, the unique number that are given to the addresses and the heading of the html pages in these addresses.

The third table that is present in the database connects the first two tables to provide a relation between the topic and the address. The Topic & Address table has two fields:

1. Topic ID
2. Address ID.

one for the topic number and the other for the address number. This then connects the topics and hence the respective pages can be accessed.

The various relations that are there in these tables are as follows:

1. Between the Topic ID of the Topic table and the Topic & Address table; a one to many relationship.
2. Between the Address ID of the Address table and the Topic & Address table; a one to many relationship.

Referential integrity has been kept in the relations so that no data that is entered in the Topic & Address table becomes redundant. Hence every time the supervisor updates the Topic & Address table he has to ensure that there are corresponding entries for the data in the Topic and Address tables.

To update the database the supervisor is provided with three applets that are connected to the database. He can use these and update the database whenever fresh data is put in the library.

THE DEMI! LIBRARY

This software comes along with the library of reference materials in which we have materials for professionals, who are not interested in the unwanted materials which are present in large amounts in the Internet. These materials will be present in HTML files in the web server.

The library is organised as follows;

Every concept has a main page where the complete links are provided for the topics that are related to the concept and that are regarding the explanations of the topic. The user can click on the hyperlinks and surf the Intranet to read and understand the topics.

The address of the reference materials will be stored in the database. The Search Engine Demi! will search for a particular reference material's address in the database using the name of the topic given to it. The library can be updated by adding new HTML pages to the web server and their corresponding address in the database. If we open a library material from that page we go to other pages of same topic using the hyperlinks in that page, but we cannot go to other main topics unless there is an option for it.

The main advantage of this library is that it avoids lot of unwanted materials entering into the web server. This library will also act like an on-line help for the all the users irrespective of their topics.

The following steps can do updating the library and providing access to the user:

1. Create the HTML page that has the reference material using Microsoft FrontPage or any HTML page creating software.

2. Add the new page to the Home Page of the corresponding concept.

3. Open the Topic table of the project database and add the new topic and provide it a new identity.

4. Now open the Address table and add the new address to the table.

5. Open the Topic & Address table and make the corresponding reference so that the topic and the addresses are connected.

The next time the user logs on to the system he can view the fresh data by accessing it through the search window. The user can be posted a note on the latest topics that are being added to the Intranet via an icon on the main screen as to the current hot topics.

SOFTWARE USED FOR DEMI!

The main software that were used for the project:

1. Microsoft Access
2. Microsoft Visual J++ &
3. Microsoft FrontPage,

Microsoft Access:

This is a full-fledged RDBMS software which is used to create databases and manage them. In the database the concepts of RDBMS like relationships, primary key, foreign key, referential integrity etc can be provided for the fields in the tables of the database.

We have created a database namely: Project. This database has 3 tables. The main criteria to use three tables was to normalise the tables and remove data redundancy. The normalisation was done with the following steps:

1. Since topics can occur a number of times it was given a unique number and split to another table.

2. Similarly the addresses were also split into another table to remove redundancy

Thus the tables that were used were:

1. Topic
2. Address
3. Topic & Address Tables.

Microsoft Visual J++:

This is an object-oriented software that has all the features including: Classes, Objects, Inheritance, Code reusability etc.

The Microsoft Visual J++ provides a graphical environment that helps the user to write source code and compile it without going to the DOS prompt. The editor also has a lot of additional features such as Wizards, OLE&COM Viewers, Project Builder, Workspace Manager etc.

The Java Resource Wizard creates the skeleton code that is necessary for applets and applications. This skeletal code can be used by the programmer to decrease coding time and also some simple errors.

The Workspace manager helps in keeping track of all the files that are being used by the programmer in a software. It provides an easy access to view and work around with all the files with the help of windows which keep track of the various procedures.

The Project Builder has options for building class files and executing a program from the environment without going to the DOS prompt. There are debuggers which help the user to debug a program on a separate window while running a program.

This being a Microsoft product there are a few extra class files that have been included for the database options. These files have to be included along with the project files for the successful running of the program. Visual J++ has all the features that are present in the Sun's Java Developer's Kit.

Microsoft uses its Data Access Objects technology to make a connection to the database. It has introduced a new set of class files that help in accessing the database. These class files are present in the c:\java directory when Visual J++ is installed in the system.

When sending an applet across the Intranet using the DAO method it is essential to send these new class files along with the applet also so that the host system can download the software and run it without any hitch.

There are digital signatures also that are needed to make these applets trusted applets. Only users who can provide a digital key to the software to authenticate the user and the system can use the trusted applets. This is essential for the organization as the data that are being sent across the Intranet should not be accidentally or purposefully read by others.

The main use of this software was to create the Search Engine. Using this software we can access the database also so that any changes that have to be made can be made to it.

The flow of the Search Engine is as follows:

i. The database is opened and a hashtable is created for two of the tables.

ii. This helps in retrieving the data at faster speed.

iii. The Search box and the screen are then enabled.

iv. The user can enter his query to the database using the Search box or the Search window.

v. The search is then checked for validity and then compared with the topics present in the database.

vi. If the topic is present then the addresses are retrieved and presented to the user in a list box.

vii. The user can then select the specific topic from the list box by double clicking on it.

Microsoft FrontPage:

This software is a HTML page creation software that helps create HTML pages with a WYSIWYG concept. It has elegant menu driven and drag and drop methods that can be used to add or update a HTML page

The FrontPage has an Explorer that helps to keep track of the web pages that are created by the user for a particular server. It also helps to

install a local web server when no server is present for the user to work with. The Explorer uses a graphical interface to show the index page and the hyperlinks that are connected to the page.

The FrontPage has an editor that has a set of pull down menus and a drag drop toolbox. The editor has al the standard features that include font selectors, text alignment, thesaurus, spell checker etc.

This software was used to create the library pages for the project. The basic format for the creation of the library pages was to have a page heading and the provide the contents below it and then provide hyperlinks wherever related words appear. At the end of the content of the page we also provide hyperlinks to the related topics that come under the title of See Also.

The addition of Images, Sound, Video clips etc can be done using the various menu driven options that are present in the software.

The various other functions that can be done are adding background images, background sound, icons, animated icons etc.

FLOW OF THE SOFTWARE

This software basically consists of search engine named Demi! which is used to search for reference materials in the library which comes along with the software. We will open this software in the browser, from there we will Search for the reference materials. This search engine Demi! consists of a main screen which has a search screen with a search box..

The topic to be referred is first entered in the search box, then after the searching process is over we will get the address or location particular topic's reference materials .If the reference material for the topic is found then that will be opened in the browser. This software starts with a search screen from there the required material is retrieved from the electronic library.

After opening the reference material we can move around the reference material using the hyperlinks provided in the library pages.

This will make user to use the library in an efficient way by reducing the access time of the pages, referring different sub topics in the same time.

The library provides hyperlinks to other topics in required places but we cannot go from one topic to topic from the library. If we want go new again we should call the search engine Demi!.

Searching by Demi! is performed in the database of addresses for the library of reference materials. This database consists three tables, first table consists of topic and topic identifier the second table consists address and topic identifier and the third table used to link the first and second table of the database. The final result i.e the location of the reference material will be obtained from third table.

Demi! is having an Online help for using it. Online help consists of various topics and it can be opened at any point when it is needed. The help consists of information about the library and how to use the library. It shows how to browse the library pages and move from one page to another page using the Hyperlinks provided in that page.

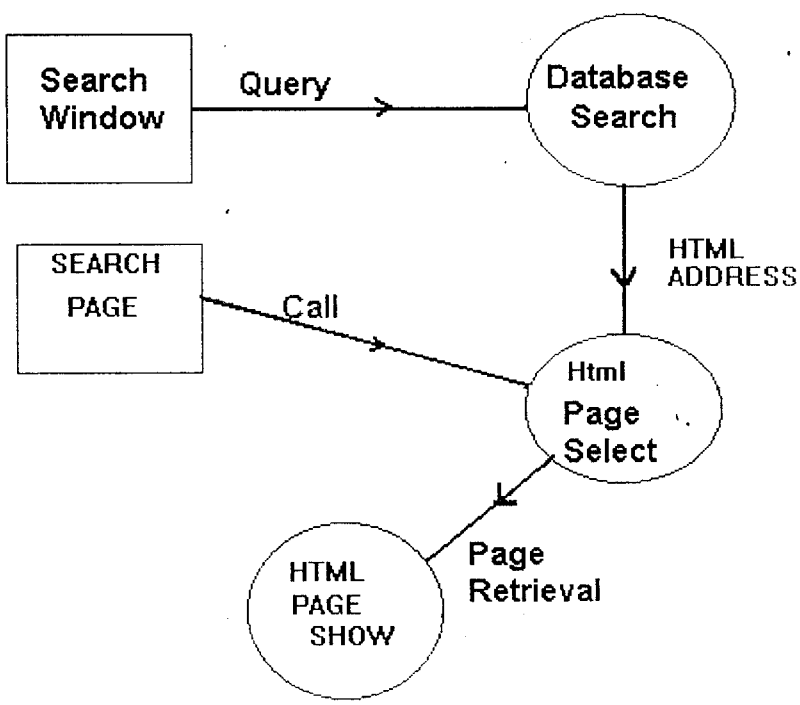
In this we have the information about the Demi! ,which is used to search the library materials .Demi! is having a search box where we have to enter the topic which we would like to refer, then it will give the address of

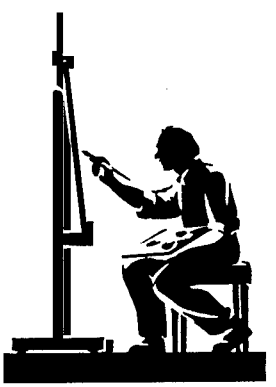
the page if it find's its address in the database we can go to that page by clicking its address.

In this we will get the information about the tips and why we need the tips for this software .It also tells about the contents of the tips. Tips will be accessed only by supervisor of the software to use and maintain the Demi!. The tips in this software is used by supervisor of this software to use this software and to maintain it. The access is denied to ordinary users by having a password for the supervisor area. The reason why we need tips for the supervisor of the software is:

* Tips of this software is accessed only by the supervisor of this software.
* Tips consists of various topics to help the supervisor
* Tips is used to maintain the library of the software and to keep updating it.
- Tips is used to maintain the database of the software.

In this software all the tips and the help are in the HTML files ,so it easy to access from any location. we need to load the help in each and every system instead of that we can have it the web server and access it whenever the user is need of help or the supervisor wants to do the maintenance job.


Search Window
Query
Database Search
HTML ADDRESS
SEARCH PAGE
Call
Html Page Select
HTML PAGE SHOW
Page Retrieval



SOURCE CODE

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
<title>HELPS</title>
</head>

<body bgcolor="#008080" topmargin="26">

<p><input type="button" name="B1" value="ABOUT"
language="VBScript"
onclick="Window.location.href = 'http://krish.edu/help/about.htm'">
</p>

<p><input type="button" name="B2" value="TIPS"
language="VBScript"
onclick="Window.location.href = 'http://krish.edu/help/tips.htm'">
</p>

<p><input type="button" name="B3" value="HOME"> </p>

<p><input type="button" name="B4" value="LIBRARY"
language="VBScript"
onclick="Window.location.href = 'http://krish.edu/help/library.htm'">
</p>

<p><input type="button" name="B5" value="SEARCH"
language="VBScript"
onclick="Window.location.href = 'http://krish.edu/help/search.htm'">
</p>
</body>
</html>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
<title>C ++</title>
</head>

<body topmargin="0">

<p align="center"><font color="#FF0000" size="4"><em><strong><u><img
src="File12A4.gif" width="30" height="32">C++
Concept</u></strong></em></font></p>

<hr width="640">

<p align="left"><font color="#000000">Basically<em><strong>
</strong></em></font><font
color="#FF0000"><em><strong>C</strong></em></font><font
color="#0000FF"><em><strong> </strong></em></font><font
color="#000000">and </font><font
color="#FF0000"><em><strong>C++</strong></em></font><font
color="#000000"><em><strong> </strong></em>are entirely seperate
languages .It's true that their syntax is similar;</font></p>

<p align="left"><font
color="#FF0000"><em><strong>C</strong></em></font><font
color="#000000"><em><strong> </strong></em>is subset of </font><font
color="#FF0000"><em><strong>C++</strong></em></font><font
color="#0000FF"><em><strong> </strong></em></font><font
color="#000000">, the similarity is just an accident.The basic
approach in a </font><font color="#FF0000">C++</font><font
color="#000000"> program</font></p>

<p align="left"><font color="#000000">is radically different from
that in a </font><font
color="#FF0000"><em><strong>C</strong></em></font><font
```

```
color="#000000"><em><strong> </strong></em>program.</font></p>

<p align="center"><font
color="#FF0000"><em><strong><u>CONTENTS</u></strong></em></fon
t></p>

<ul>
   <li><p align="left"><a href="whyoops.htm"><font
      color="#000000"><em><strong><u>Why
Oops</u></strong></em></font></a></p>
   </li>
   <li><p align="left"><a
href="programm.htm"><em><strong><u>Programming
      Basics</u></strong></em></a></p>
   </li>
   <li><p align="left"><a href="objects.htm"><em><strong><u>Objects
      &amp;amp; Classes</u></strong></em></a></p>
   </li>
   <li><p align="left"><a href="operator.htm"><em><strong><u>Operator
      Overloading</u></strong></em></a></p>
   </li>
   <li><p align="left"><a
href="inherita.htm"><em><strong><u>Inheritance</u></strong></em></a><
em><strong><u>
      </u></strong></em></p>
   </li>
   <li><p align="left"><a
href="pointers.htm"><em><strong><u>Pointers</u></strong></em></a></p
>
   </li>
   <li><p align="left"><a href="special.htm"><em><strong><u>Special
      Features</u></strong></em></a></p>
   </li>
</ul>

<ul>
   <li><a href="virtual.htm"><em><strong><u>Virtual
Functions</u></strong></em></a></li>
</ul>
```

```
<hr width="640">

<p align="left"><font color="#00FF00"><em><strong><u>See
Also</u></strong></em></font></p>

<p align="left"><font
color="#00FF00"><em><strong><u>tyuuuu</u></strong></em></font></p
>

<p align="left"><font
color="#00FF00"><em><strong><u></u></strong></em></font> </p
>
</body>
</html>
```

```
<html>
<head>
<title>"Address</title>
</head>
<body>
<font size=5>"Address" table of "project" database</font>
<hr>
<applet
   code=Address.class
   id=Address
   width=600
   height=312>
</applet>
<hr>
<a href="Address.java">The source.</a>
</body>
</html>
```

```
<html>
<head>
<title>"Search</title>
</head>

<body background = "Image\bkg1.gif">
<h1>Search Engine for an Intranet</h1>
<hr>
<p><applet code="Search.class" align="baseline" width="550"
height="200" id="Search"></applet></p>

<hr>

<p><a href="file:h:\library\oops\oops.html"><font
size="4"><b>Oops</b></font></a><font
size="4"> </font></p>
</body>
</html>
```

```
import java.applet.*;
import java.awt.*;
import java.util.*;
import dao350.*;
import com.ms.com.*;
import Alert;
import java.net.*;

class DBField
{
   String strName;
   String strType;
   short vt;
   Label label;
   TextField field;
   int attributes;
   String strValidation;
   public DBField(String strName, String strType, short vt)
   {
      this.strName = strName;
      this.strType = strType;
      this.vt = vt;
   }
}

class DBFrame extends Frame
{
   Search applet;

   public DBFrame(Search appletSearch, String strTitle)
   {
      super(strTitle);
      this.applet = appletSearch;
   }

   public synchronized boolean handleEvent(Event evt)
   {
      switch (evt.id)
      {
         case Event.WINDOW_DESTROY:
            applet.exit();
```

```
            return true;

        default:
            return super.handleEvent(evt);
      }
   }
}

public class Search extends Applet
{
   static protected Search applet = null;
   protected Frame frame;
   protected boolean readOnly = true;
   protected String strDatabase = "H:\\database\\project.mdb";
   protected String strRecordset = "Topic";
       protected String strRecordset1 = "Address";
       protected String strRecordset2 = "TopAdd";
   protected int recordCount;
   protected int recordCount1;
   protected int recordCount2;
       protected static Variant varEmpty;
       protected _DBEngine m_IEngine;
   protected Database database;
   protected Recordset recordset;
   protected Recordset recordset1;
   protected Recordset recordset2;

   static DBField[] field =
   {
      new DBField( "Topic",   "Text", Variant.VariantString   ),
            new DBField( "TopicId", "Long", Variant.VariantInt      ),
   };

   static DBField[] field1 =
   {
      new DBField( "AddressId", "Long", Variant.VariantInt      ),
      new DBField( "Address",   "Text", Variant.VariantString   ),
            new DBField( "AddTopic",  "Text", Variant.VariantString   ),
   };

    static DBField[] field2 =
```

```
{
    new DBField( "TopicId",   "Long", Variant.VariantInt      ),
            new DBField( "AddressId", "Long", Variant.VariantInt      ),

};

      Hashtable hashField = new Hashtable();


      Hashtable hashField1 = new Hashtable();

Hashtable hashField2 = new Hashtable();


  protected int columns = 1;
  protected int textFieldWidth = 25;

  Variant vName;
  Variant vValue;
  Variant vOriginalValue;

     protected Panel feedback = new Panel();
      protected Button buttonSearch = new Button("Search");
      protected Label labelSearch = new Label("Enter the search text");
      protected TextField textSearch = new TextField("",30);
      protected Panel listpage = new Panel();
      protected Label labelList = new Label("Double click below to read
the page");
      protected List listSearch = new List(10,false);
      Image bkg;
      String add;
  String topic;
      String Addtopic;
      protected void makeFrame()
  {
    frame = new DBFrame(this, "Search");
  }

  public static void main(String[] args)
  {
    applet = new Search();
```

```
    applet.makeFrame();
            applet.frame.add("Center", applet);
            applet.init();
            applet.start();
            applet.frame.pack();
            applet.frame.show();
            applet.frame.setResizable(false);

}

public String getAppletInfo()
{
    return "Name: Search\n" +
        "Author: t.c.satheesh\n" +
        "Created with Microsoft Visual J++ Version 1.1";
}

     public void paint(Graphics g)
     {
            g.drawImage(bkg,0,0,this);

     }
public void init()
{

            varEmpty = new Variant();
            varEmpty.putEmpty();

    ILicenseMgr mgr = new LicenseMgr();
    m_IEngine = (_DBEngine)mgr.createWithLic
    (
                    "mbmabptebkjcdlgtjmskjwtsdhjbmkmwtrak",
                    "{00000010-0000-0010-8000-00AA006D2EA4}",
       null,
       ComContext.INPROC_SERVER
    );

    Variant vExclusive = new Variant();
    Variant vReadOnly = new Variant();
    Variant vConnect = new Variant();
    vExclusive.putBoolean(false);
```

```
        vReadOnly.putBoolean(readOnly);
        vConnect.putString("");
        database = m_IEngine.OpenDatabase(strDatabase, vExclusive,
vReadOnly, vConnect);

        Variant vOpenType = new Variant();
        Variant vOptions = new Variant();
                Variant vLockType = new Variant();
        vOpenType.putShort((short)RecordsetTypeEnum.dbOpenDynaset);
        vOptions.putShort((short)0);
                vLockType.putInt(readOnly ?
RecordsetOptionEnum.dbReadOnly : LockTypeEnum.dbOptimistic);
        recordset = database.OpenRecordset(strRecordset, vOpenType,
vOptions, vLockType);
                recordset1 = database.OpenRecordset(strRecordset1,
vOpenType, vOptions, vLockType);
                recordset2 = database.OpenRecordset(strRecordset2,
vOpenType, vOptions, vLockType);

                if (recordset.getEOF())
                        recordCount = 0;
                else
                {
                        int nOptions = 0;
                        recordset.MoveLast(nOptions);
                        recordCount = recordset.getRecordCount();
                        recordset.MoveFirst();
                }


                if (recordset1.getEOF())
                        recordCount1 = 0;
                else
                {
                        int nOptions = 0;
                        recordset1.MoveLast(nOptions);
                        recordCount1 = recordset1.getRecordCount();
                        recordset1.MoveFirst();
                }

                if (recordset2.getEOF())
```

```
                recordCount2 = 0;
            else
            {
                int nOptions = 0;
                recordset2.MoveLast(nOptions);
                recordCount2 = recordset2.getRecordCount();
                recordset2.MoveFirst();
            }


            vName = new Variant();
        vValue = new Variant();
        vOriginalValue = new Variant();


        feedback.setLayout(new BorderLayout());
                listpage.setLayout(new BorderLayout());
                feedback.add("West",labelSearch);
                feedback.add("Center",textSearch);
                feedback.add("East",buttonSearch);
                listpage.add("North",labelList);
                listpage.add("South",listSearch);
                setBackground(Color.white);
                add("North",feedback);
                add("Center",listpage);
                listSearch.setBackground(Color.white);
                listSearch.setForeground(Color.black);
                labelSearch.setBackground(Color.blue);
                labelSearch.setForeground(Color.white);
                buttonSearch.setBackground(Color.blue);
                buttonSearch.setForeground(Color.white);

        try
                {
              bkg=getImage(new URL
("file:h:\\project\\Search\\Image\\new.gif"));
            }catch (MalformedURLException ex){
                }
                hashing();
                hashing1();
```

```
            hashing2();
        }

    public void destroy()
    {
    }
        public boolean searchtopadd(String topicid)
        {
                String Addid,Topid;
                recordset2.MoveFirst();
                vName.putString(field2[0].strName);
                vValue = recordset2.getCollect(vName);
                Topid = vValue.toString();
                vName.putString(field2[1].strName);
                vValue = recordset2.getCollect(vName);
                Addid = vValue.toString();

                int i=1;
                do
                {
                        if(Topid.equals(topicid))
                        {

                                listSearch.addItem((String)
hashField2.get(Addid));
                        }

                                recordset2.MoveNext();
                                vName.putString(field2[0].strName);
                                vValue = recordset2.getCollect(vName);
                                Topid = vValue.toString();
                                vName.putString(field2[1].strName);
                                vValue = recordset2.getCollect(vName);
                                Addid = vValue.toString();


                                i++;
                }while (i <= recordCount2+1);
                        return false;
        }

        public void hashing()
```

```
  {
        String dbtext,dbvalue;
        recordset.MoveFirst();
for (int i = 1; i < recordCount+1; i++)
 {

                    vName.putString(field[0].strName);
                    vValue = recordset.getCollect(vName);
                    dbtext = vValue.toString();
                    vName.putString(field[1].strName);
                    vValue = recordset.getCollect(vName);
                    dbvalue = vValue.toString();
      hashField.put(dbtext,dbvalue);
                    recordset.MoveNext();

 }
  }



  public void hashing1()
  {
        String dbtext,dbvalue;
        recordset1.MoveFirst();
for (int i = 1; i < recordCount1+1; i++)
 {

                    vName.putString(field1[2].strName);
                    vValue = recordset1.getCollect(vName);
                    dbtext = vValue.toString();
                    vName.putString(field1[1].strName);
                    vValue = recordset1.getCollect(vName);
                    dbvalue = vValue.toString();
      hashField1.put(dbtext,dbvalue);
                    recordset1.MoveNext();

 }
  }

  public void hashing2()
```

```
    {
            String dbtext,dbvalue;
            recordset1.MoveFirst();
    for (int i = 1; i < recordCount1+1; i++)
     {

                        vName.putString(field1[0].strName);
                        vValue = recordset1.getCollect(vName);
                        dbtext = vValue.toString();
                        vName.putString(field1[2].strName);
                        vValue = recordset1.getCollect(vName);
                        dbvalue = vValue.toString();
          hashField2.put(dbtext,dbvalue);
                        recordset1.MoveNext();

     }
      }


            public boolean searching()
     {

            String topicid="";
                  topicid= (String) hashField.get(topic);
                  if(!topicid.equals(""))
                  {

                        if(searchtopadd(topicid))
                        return true;
                  }
            return false;

     }
  public synchronized boolean action(Event evt, Object o)
  {
     if (evt.target == buttonSearch)
     {
                        listSearch.clear();
                        StringTokenizer txtbox = new
StringTokenizer(textSearch.getText());
                        topic = txtbox.nextToken();
```

```
                        searching();
                }
                else if (evt.target == listSearch)
                {
                        try
                        {
                                Addtopic=listSearch.getSelectedItem();
                                add ="file:"+(String)
hashField1.get(Addtopic);
                                URL a = new URL(add);
                                getAppletContext().showDocument(a);
                                return true;
                        }catch(Exception e){
                                new Alert(frame,"URL error","change it");
                        }
        }
        return false;
    }

    public void start()
    {
                repaint();
    }

    public synchronized void stop()
    {
        if (frame != null)
            System.exit(0);
    }

 protected void exit()
    {
        if (recordset != null)
        {
            recordset.Close();
            recordset = null;
        }
        stop();
    }

 }
```

```
import java.applet.*;
import java.awt.*;
import java.util.*;
import dao350.*;
import com.ms.com.*;
import Alert;

class DBField
{
   String strName;
   String strType;
   short vt;
   Label label;
   TextField field;
   int attributes;
   String strValidation;
   public DBField(String strName, String strType, short vt)
   {
      this.strName = strName;
      this.strType = strType;
      this.vt = vt;
   }
}

class DBFrame extends Frame
{
   Address applet;

   public DBFrame(Address appletAddress, String strTitle)
   {
      super(strTitle);
      this.applet = appletAddress;
   }

   public synchronized boolean handleEvent(Event evt)
   {
      switch (evt.id)
      {
         case Event.WINDOW_DESTROY:
            applet.exit();
            return true;
```

```
            default:
               return super.handleEvent(evt);
         }
      }
   }

public class Address extends Applet
{
   static protected Address applet = null;
   protected Frame frame;
   protected boolean readOnly = false;
   protected String strDatabase = "F:\\database\\project.mdb";
   protected String strRecordset = "Address";
   protected int recordCount;
        protected static Variant varEmpty;
        protected _DBEngine m_IEngine;
   protected Database database;
   protected Recordset recordset;

   static DBField[] field =
   {
      new DBField( "Address",   "Text", Variant.VariantString   ),
      new DBField( "AddressId", "Long", Variant.VariantInt      ),
      new DBField( "AddTopic",  "Text", Variant.VariantString   ),
   };

   static Hashtable hashField = new Hashtable();

   static
   {
      for (int i = 0; i < field.length; i++)
      {
         hashField.put(field[i].strName, field[i]);
      }
   }

   protected int columns = 1;
   protected int textFieldWidth = 25;

   Variant vName;
```

```
Variant vValue;
Variant vOriginalValue;

protected Panel db = new Panel();
protected Panel dbcolumn[] = new Panel[columns * 2];

protected Panel tools = new Panel();
protected Panel toolbar = new Panel();
protected Button buttonFirst = new Button("<< First");
protected Button buttonPrev = new Button("< Prev");
protected Button buttonNext = new Button("Next >");
protected Button buttonLast = new Button("Last >>");

protected Panel updatetools = new Panel();
protected Button buttonUpdate = new Button("Save!");
protected Button buttonAdd = new Button("Add!");
protected Button buttonDelete = new Button("Delete!");
protected Button buttonReread = new Button("Re-read");
protected Button buttonExit = new Button("Exit");

protected Panel feedback = new Panel();
protected Label labelDatabase = new Label("", Label.CENTER);
protected Label labelRecordset = new Label("", Label.CENTER);
protected Label labelPosition = new Label("", Label.CENTER);
    protected Label labelStatus = new Label("
");

protected void makeFrame()
{
  frame = new DBFrame(this, "Address");
}

public static void main(String[] args)
{
  applet = new Address();
  applet.makeFrame();
        applet.frame.add("Center", applet);
        applet.init();
        applet.start();
        applet.frame.pack();
        applet.frame.show();
```

```
            applet.frame.setResizable(false);
    }

    public String getAppletInfo()
    {
        return "Name: Address\n" +
            "Author: Unknown\n" +
            "Created with Microsoft Visual J++ Version 1.1";
    }

    public void init()
    {
            varEmpty = new Variant();
            varEmpty.putEmpty();

        ILicenseMgr mgr = new LicenseMgr();
        m_IEngine = (_DBEngine)mgr.createWithLic
        (
                    "mbmabptebkjcdlgtjmskjwtsdhjbmkmwtrak",
                    "{00000010-0000-0010-8000-00AA006D2EA4}",
          null,
          ComContext.INPROC_SERVER
        );

        Variant vExclusive = new Variant();
        Variant vReadOnly = new Variant();
        Variant vConnect = new Variant();
        vExclusive.putBoolean(false);
        vReadOnly.putBoolean(readOnly);
        vConnect.putString("");
        database = m_IEngine.OpenDatabase(strDatabase, vExclusive,
vReadOnly, vConnect);

        Variant vOpenType = new Variant();
        Variant vOptions = new Variant();
            Variant vLockType = new Variant();
        vOpenType.putShort((short)RecordsetTypeEnum.dbOpenDynaset);
        vOptions.putShort((short)0);
            vLockType.putInt(readOnly ?
RecordsetOptionEnum.dbReadOnly : LockTypeEnum.dbOptimistic);
```

```
    recordset = database.OpenRecordset(strRecordset, vOpenType,
vOptions, vLockType);

            if (recordset.getEOF())
                recordCount = 0;
            else
            {
                int nOptions = 0;
                recordset.MoveLast(nOptions);
                recordCount = recordset.getRecordCount();
                recordset.MoveFirst();
            }
        vName = new Variant();
        vValue = new Variant();
        vOriginalValue = new Variant();

        columns *= 2;
        db.setLayout(new GridLayout(0, columns));
        for (int col = 0; col < columns; col++)
        {
          db.add(dbcolumn[col] = new Panel());
          dbcolumn[col].setLayout(new GridLayout(0, 1));
        }

        int col = 0;
        Fields fields = recordset.getFields();
        for (int i = 0; i < field.length; i++)
        {
          DBField df = (DBField)hashField.get(field[i].strName);
          vName.putString(field[i].strName);
          _Field thisfield = fields.getItem(vName);
          df.strValidation = thisfield.getValidationText();
          df.attributes = thisfield.getAttributes();
          String attribs = new String();
          df.field = new TextField(textFieldWidth);
          if (!readOnly && ((df.attributes &
FieldAttributeEnum.dbUpdatableField) == 0))
          {
            attribs += " (ReadOnly)";
            df.field.disable();
          }
```

```
    if ((df.attributes & FieldAttributeEnum.dbAutoIncrField) != 0)
    {
       attribs += " (Auto)";
       df.field.disable();
    }
    df.label = new Label
    (
       df.strName + " (" + df.strType + ")" + attribs + " :",
       Label.RIGHT
    );
    if (readOnly)
    {
       df.field.disable();
    }
    dbcolumn[col++].add(df.label);
    dbcolumn[col++].add(df.field);
    col %= columns;
}
toolbar.setLayout(new GridLayout(1, 0));
toolbar.add(buttonFirst);
toolbar.add(buttonPrev);
toolbar.add(buttonNext);
toolbar.add(buttonLast);

updatetools.setLayout(new GridLayout(1, 0));
updatetools.add(buttonUpdate);
updatetools.add(buttonAdd);
updatetools.add(buttonDelete);
updatetools.add(buttonReread);

tools.setLayout(new FlowLayout());
tools.add(toolbar);
if (!readOnly)
{
   tools.add(updatetools);
}
          if (frame != null)
        tools.add(buttonExit);

feedback.setLayout(new GridLayout(0, 1));
feedback.add(labelDatabase);
```

```
    feedback.add(labelRecordset);
    feedback.add(labelPosition);
            feedback.add(labelStatus);

    GridBagLayout gb = new GridBagLayout();
    setLayout(gb);
    GridBagConstraints c = new GridBagConstraints();
    c.insets = new Insets(20, 30, 20, 30);
    c.gridx = 0;
    add(feedback);
    gb.setConstraints(feedback, c);
    add(db);
    gb.setConstraints(db, c);
    add(tools);
    gb.setConstraints(tools, c);

    updateUI();
}

public void destroy()
{
}

public void start()
{
}

public synchronized void stop()
{
    if (frame != null)
        System.exit(0);
}

public synchronized boolean action(Event evt, Object o)
{
            labelStatus.setText("");

    if (evt.target == buttonFirst)
    {
        recordset.MoveFirst();
        updateUI();
```

```
    return true;
}
else if (evt.target == buttonPrev)
{
    recordset.MovePrevious();
    updateUI();
    return true;
}
else if (evt.target == buttonNext)
{
    recordset.MoveNext();
    updateUI();
    return true;
}
else if (evt.target == buttonLast)
{
                int nOptions = 0;
    recordset.MoveLast(nOptions);
    updateUI();
    return true;
}
else if (evt.target == buttonUpdate)
{
    buttonUpdate.disable();
    buttonAdd.disable();
    buttonUpdate.disable();
    updateDatabase(EditModeEnum.dbEditInProgress);
    buttonUpdate.enable();
    buttonAdd.enable();
    buttonDelete.enable();
    return true;
}
else if (evt.target == buttonAdd)
{
    buttonUpdate.disable();
    buttonAdd.disable();
    buttonUpdate.disable();
    updateDatabase(EditModeEnum.dbEditAdd);
    buttonUpdate.enable();
    buttonAdd.enable();
    buttonDelete.enable();
```

```
        return true;
    }
    else if (evt.target == buttonDelete)
    {
        buttonUpdate.disable();
        buttonAdd.disable();
        buttonUpdate.disable();
        updateDatabase(EditModeEnum.dbEditNone);
        buttonUpdate.enable();
        buttonAdd.enable();
        buttonDelete.enable();
        return true;
    }
    else if (evt.target == buttonReread)
    {
        updateUI();
        return true;
    }
    else if (evt.target == buttonExit)
    {
        exit();
    }
    return false;
}

protected void exit()
{
    if (recordset != null)
    {
        recordset.Close();
        recordset = null;
    }
    stop();
}

void updateTextField(DBField f)
{
    if (recordset != null)
    {
        String strValue;
        try
```

```
        {
            vName.putString(f.strName);
            Variant vValue;
            vValue = recordset.getCollect(vName);
            if (f.vt == Variant.VariantBoolean)
            {
                Boolean b = new Boolean(vValue.getBoolean());
                strValue = b.toString();
            }
            else
            {
                strValue = vValue.toString();
            }
        }
        catch (ClassCastException c)
        {
            strValue = "";
        }
        f.field.setText(strValue);
    }
}

void updateDatabaseField(DBField df) throws Exception
{
    if (recordset != null)
    {
        try
        {
            String strValue = df.field.getText();
            vValue.putString(strValue);
            vName.putString(df.strName);
            if (df.vt == Variant.VariantBoolean)
            {
                if (strValue.equals("true"))
                {
                    vValue.putBoolean(true);
                }
                else
                if (strValue.equals("false"))
                {
                    vValue.putBoolean(false);
```

```
                }
                else
                {
                    throw new Exception(df.strName + " must be either 'true' or
'false' in lowercase.");
                }
            }
            else
            {
                vValue.changeType(df.vt);
            }
            vOriginalValue = recordset.getCollect(vName);
            if (!((vOriginalValue.getvt() == Variant.VariantNull) &&
                (strValue.equals(""))))
            {
                recordset.putCollect(vName, vValue);
            }
        }
        catch (ClassCastException c)
        {
        }
    }
}

protected String getExceptionMessage(Exception e, DBField f)
{
    if (e instanceof ComFailException)
    {
        int h = ((ComFailException)e).getHResult();
        switch (h)
        {
        case 0x800a0c5b:
            return "Field value is too long";

        case 0x800a0cf4:
            if ((f == null) || f.strValidation.equals(""))
            {
                return "Validation rule failed";
            }
            else
            {
```

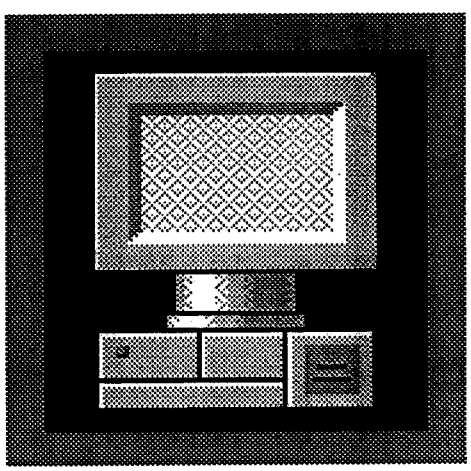
```
                return f.strValidation;
            }

        default:
            return "DAO COM Exception " + e.getMessage();
        }
    }
    return e.getMessage();
}

protected void updateDatabase(int nMode)
{
    try
    {
                    if (nMode == EditModeEnum.dbEditNone)
                    {
                            recordset.Delete();

                            recordset.MoveNext();
                            if (recordset.getEOF())
                                    recordset.MovePrevious();
                            recordCount--;
                    }
                    else
                    {
                            if (nMode == EditModeEnum.dbEditAdd)
                                    recordset.AddNew();
                            else
                                    recordset.Edit();

                            int i;
                            for (i = 0; i < field.length; i++)
                            {
                      if (((field[i].attributes &
FieldAttributeEnum.dbUpdatableField) != 0) &&
                                ((field[i].attributes &
FieldAttributeEnum.dbAutoIncrField) == 0))
                                        {
                                                try
                                                {
                                        updateDatabaseField(field[i]);
```

```
                        }
                            catch (Exception e)
                                {
                                        String strError;
                                        strError = "Error updating " +
field[i].strName + " : " + getExceptionMessage(e, field[i]);
                                        if (frame != null)
                                                new Alert(frame,
"Modification Failed!", strError);
                                        else

        labelStatus.setText(strError);

recordset.CancelUpdate(UpdateTypeEnum.dbUpdateRegular);
                        break;
                          }
                            }
                }
                    if (i == field.length)
                        {
                   recordset.Update(UpdateTypeEnum.dbUpdateRegular,
false);
                                if (nMode == EditModeEnum.dbEditAdd)
                                {
                                        recordCount++;

                                        if (recordset.getEOF())
                                                recordset.MoveFirst();
                                }
                        }
                    }
                }
                catch (ComException eCom)
                {
                        String strError = new String();
                        Errors errs = m_IEngine.getErrors();
                        Variant var = new Variant();

                        for (int n = 0; n < errs.getCount(); n++)
                        {
                                var.putInt(n);
```

```
                        dao350.Error err = errs.getItem(var);
                        if (nMode == EditModeEnum.dbEditAdd)
                                strError += "Error adding record: ";
                        else if (nMode ==
EditModeEnum.dbEditInProgress)
                                strError += "Error updating record: ";
                        else
                                strError += "Error deleting record: ";

                        strError += err.getDescription();
                }

                if (frame != null)
                        new Alert(frame, "Modification Failed!",
strError);
                else
                        labelStatus.setText(strError);
            }
        catch (Exception e)
        {
                String strError;
                strError = "Error updating record: " +
getExceptionMessage(e, null);
                if (frame != null)
                        new Alert(frame, "Modification Failed!",
strError);
                else
                        labelStatus.setText(strError);
        }
        updateUI();
    }

    protected void updateUI()
    {
            int pos = -1;
            if (recordCount > 0)
            {
                for (int i = 0; i < field.length; i++)
                {
                    try
                    {
```

```
                        updateTextField(field[i]);
                    }
                    catch (Exception e)
                    {
                    String strError;
                    strError = "Error retrieving field value for " +
field[i].strName + " : " + e.getMessage();
                    if (frame != null)
                        new Alert(frame, "Data Fetch Failed!",
strError);
                    else
                        labelStatus.setText(strError);
                    break;
                }
                }

                    pos = recordset.getAbsolutePosition();
                }

        buttonFirst.enable(pos != 0 && recordCount > 0);
        buttonPrev.enable(pos != 0 && recordCount > 0);
        buttonNext.enable(pos != (recordCount - 1));
        buttonLast.enable(pos != (recordCount - 1));
        labelDatabase.setText("Database: " + strDatabase + (readOnly ? "
(Read Only)" : ""));
        labelRecordset.setText("Table: " + strRecordset);
        labelPosition.setText("Record: " + (pos + 1) + " of " + recordCount);
            if (recordCount == 0)
                labelStatus.setText("There are no records in the table!");
    }
}
```

```
import java.awt.*;

public class Alert extends Dialog
{
   protected Button buttonOK;

   public Alert(Frame parent, String title, String text)
   {
      super(parent, title, true);
      GridBagLayout gb = new GridBagLayout();
      setLayout(gb);
      GridBagConstraints c = new GridBagConstraints();
      c.gridx = 0;
      c.insets = new Insets(10, 10, 10, 10);
      Label l;
      add(l = new Label(text));
      gb.setConstraints(l, c);
      add(buttonOK = new Button("OK"));
      gb.setConstraints(buttonOK, c);
      setResizable(false);
      pack();
      Rectangle rcParent = parent.bounds();
      Dimension dAlert = size();
      move
      (
         rcParent.x + (rcParent.width - dAlert.width) / 2,
         rcParent.y + (rcParent.height - dAlert.height) / 2
      );
      show();
   }

   public boolean action(Event e, Object o)
   {
      if (e.target == buttonOK)
      {
         dispose();
         return true;
      }
      return false;
   }
}
```



SAMPLE SCREENS

FURTHER IMPROVEMENTS

The project that we have undertaken has a lot of scope of improvement. We believe that there are a few more facilities that could be provided to the supervisor and the user in the creation and maintenance of the library and the Demi! Search Engine.

A few improvements that we would like to suggest are:

1. To provide dynamic databases update software so that the supervisor does not have to do it manually. In order to achieve this the updating software must be optimised to search for pages that are relevant for the database.

2. Another improvement can be to provide a standard HTML page creator that can be used to create fresh pages to the library.

These tasks can improve the efficiency of the software much more and thus provide a better productivity for the organisation.

CONCLUSION

CONCLUSION

We have implemented the software and tested in the Computer Center of KCT with successful results. The final implementation in an organization can be accomplished by acquiring a suitable X.509 certificate and getting a suitable digital signature via the Internet and then providing a unique digital key to the organization and its employees so that only the employees can access the data. Hence security for the data is absolute. The security technology that is used is Microsoft's Authenticode technology.

BIBLIOGRAPHY

1. JAVA DEVELOPERS GUIDE; by, Jamie Jaworski ; Sam's Net Publishing.

2. THE JAVA SOURCE BOOK; by, Ed Anuff John Wiley & Son's Inc

3. JavaScript HANDBOOK; by, Danny Goodman.

4. Teach yourself Microsoft Access 97; by Charles Siegel, BPB Publications.

5. Visual J++ On line reference materials and help system.