

California State University, San Bernardino

**CSUSB ScholarWorks**

---

Theses Digitization Project

John M. Pfau Library

---

2004

## A book management system eLibrary

Shanpeng Song

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



Part of the [Data Storage Systems Commons](#), and the [Software Engineering Commons](#)

---

### Recommended Citation

Song, Shanpeng, "A book management system eLibrary" (2004). *Theses Digitization Project*. 31.  
<https://scholarworks.lib.csusb.edu/etd-project/31>

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](mailto:scholarworks@csusb.edu).

A BOOK MANAGEMENT SYSTEM --

ELIBRARY

---

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  
Shanpeng Song  
December 2004

A BOOK MANAGEMENT SYSTEM --  
ELIBRARY

---

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

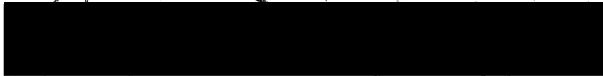
---

by  
Shanpeng Song  
December 2004

Approved by:

  
\_\_\_\_\_  
Dr. Keith Schubert, Computer Science

Nov 30, 2004  
Date

  
\_\_\_\_\_  
Dr. Owen Murphy

  
\_\_\_\_\_  
Dr. Kerstin Voigt

## ABSTRACT

As the number of books grows in one's book collection, it would be convenient to have a software utility to manage them. In this project, we propose a software system called eLibrary to manage personal book collections. eLibrary is a software application running on Microsoft® Windows® platforms. It stores book information in a local database; book information can be easily added, deleted, updated or searched in the book database. Most of the book's information, such as book title, author, publisher and cover picture, can be downloaded from the Internet; the user only needs to input one or more ISBNs.

eLibrary has been released to the public as a freeware application and it has gained much popularity among users. Many software download sites have given it the highest five-star award. A website (<http://www.elibpro.com> or <http://songstech.com>) has been set up for eLibrary, where the users can download the latest version and obtain other relevant information.

#### ACKNOWLEDGMENTS

First of all, I would like to express my special thanks to Dr. Keith Schubert, who has advised me in every aspect of this project and encouraged me to complete the project of highest value. I also express sincere thanks to Dr. Owen Murphy and Dr. Kerstin Voigt, who gave me valuable suggestions on this project.

I thank my wife and parents, who are happy with my dream to pursue higher education.

The support of the National Science Foundation under the award 9810708 is gratefully acknowledged.

## TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGMENTS.....	iv
LIST OF FIGURES.....	vii
CHAPTER ONE: INTRODUCTION	
1.1 Introduction.....	1
1.2 Purpose of this Project.....	1
1.3 Software Features.....	2
1.4 Organization of the Thesis.....	3
CHAPTER TWO: SOFTWARE REQUIREMENTS SPECIFICATION	
2.1 Introduction.....	4
2.1.1 Scope.....	4
2.2 Overall Description.....	4
2.2.1 Product Perspective.....	5
2.2.2 Product Functions.....	7
2.2.3 User Characteristics.....	10
2.3 Specific Requirements.....	11
2.3.1 External Interface Requirements.....	12
2.3.2 Functional Requirements.....	13
2.3.3 Performance Requirements.....	13
2.3.4 Software System Attributes.....	13
CHAPTER THREE: DESIGN AND IMPLEMENTATION	
3.1 Design and Implementation Decisions.....	14
3.1.1 Target Operating System.....	14

3.1.2 Data File Format.....	14
3.2 Database Design.....	15
3.2.1 Database Schema.....	15
3.2.2 Database Details.....	17
3.3 Program Design and Implementation.....	23
3.3.1 System Architecture.....	23
3.3.2 Program Details.....	27
CHAPTER FOUR: DEPLOYMENT	
4.1 System Requirements.....	37
4.2 Installation.....	37
4.2.1 Installer.....	37
4.2.2 Installation of eLibrary.....	38
CHAPTER FIVE: CONCLUSION AND FUTURE DIRECTIONS	
5.1 Conclusion.....	40
5.2 Future Directions.....	41
APPENDIX: LIST OF SOURCE CODE FILES.....	42
BIBLIOGRAPHY.....	47

## LIST OF FIGURES

Figure 1. Screenshot of the Prototype.....	6
Figure 2. the Use Case Diagram.....	8
Figure 3. Database Schema.....	16
Figure 4. Information Retrieval Process.....	24
Figure 5. Information Display Process.....	26
Figure 6. eLibrary Screenshot.....	28
Figure 7. Class Diagram of CeLibTreeView.....	31
Figure 8. Class Diagram of CeLibListView.....	33
Figure 9. Class Diagram of CeLibHtmlView.....	36
Figure 10. eLibrary Installer.....	39



## CHAPTER ONE

### INTRODUCTION

#### 1.1 Introduction

Everybody has some books; some people may have a lot. As the number of books grows in one's collection, it would be convenient to have a software utility to manage them: generate reports on the books currently in collection, keep track of book loans, quickly find the books he or she needs from the bookshelves, etc. Also, many books now include their electronic version in CD-ROMs (normally in PDF or CHM format), and some authors even offer free download of their books (such as Bruce Eckel, the author of *Thinking in C++*) [2]. Furthermore, researchers may have acquired a vast collection of electronic research documents over the years, thanks to the organizations offering full-text download (usually in PDF or PS format). People also need an efficient way to manage these electronic documents, since the file names of these documents are often meaningless and therefore it becomes extremely hard to find the exact one needed in a collection of hundreds or thousands of electronic documents.

#### 1.2 Purpose of this Project

The proposed book management system is a software application called eLibrary. The goal of this project is to

build a full-featured, commercial-quality software package to help people manage their books (either printed or electronic).

### 1.3 Software Features

At the time of this writing, the latest version of eLibrary is 1.0 RC1 and it has the following features:

- Uses a tree structure to manage book categories. It has a familiar Windows Explorer-like user interface
- Unlimited "Related Links" can be added for each book. The "Links" can be URLs, eBook files on local hard disk, or even folders
- Uses XML/XSL to display book details. Its content template and display style are completely configurable by the user
- Most of the book's information can be retrieved from the web. It can submit queries online (Amazon.com) by ISBN, or by a combination of book title, author and/or publisher
- Multiple selection (using Ctrl, Shift or Mouse) is supported in the book list window, and the book detail window can update on the fly
- Sort on any column in the book list window

- WYSIWYG editor for books' "Description" and "Notes" fields
- Drag & Drop support
- Search in the book collection
- "Favorites" and "Reading List" support
- Multi-language User Interface (MUI) support
- Native Unicode support
- Barcode scanners are supported.

#### 1.4 Organization of the Thesis

This document is organized in five chapters: (1) Introduction, (2) Software Requirements Specification, (3) Design and Implementation, (4) Deployment and (5) Technology Highlights, Conclusion and Future Development. The appendix provides a list of source code files used in this software.

## CHAPTER TWO

### SOFTWARE REQUIREMENTS SPECIFICATION

#### 2.1 Introduction

The scope of the project will be described in Section 2.1.1. Section 2.2 provides an overall description of the product while Section 2.3 details the specific requirements.

##### 2.1.1 Scope

The proposed book management system, eLibrary, is a database program that helps people catalog their books. It should be able to retrieve book information, such as book title, author and cover pictures, from the Internet based on ISBN or other information provided, so that the typing from the users will be minimal. The product should provide a friendly user interface; an average user should have no problem using it without being specially trained.

#### 2.2 Overall Description

Having so many books (many of them being ebooks), it would be nice to have a software application to manage all of them. eLibrary is such a software utility to manage the user's personal book collection (both print books and ebooks). Book information can be downloaded from the Internet; what the user needs to do is just enter one or more ISBNs, or simply use a barcode scanner.

The book information should be stored in a database. The user can work with an unlimited number of databases, which means the user can split his or her book collection into different catalogs.

#### 2.2.1 Product Perspective

The product should be independent and totally self-contained. It should not rely on any external libraries or frameworks.

2.2.1.1 System Interfaces. The system will use HTTP to retrieve book information from the Internet (such as Amazon.com); the user verifies the retrieved information and enters additional information; then the system stores all the information to a local database.

2.2.1.2 User Interfaces. The system should provide an easy-to-use user interface, using standard GUI elements such as menus, toolbars, dialogs, shortcut keys etc. In addition, a Multi-language User Interface (MUI) framework should be supported so that people from all around the world can choose their preferred interface languages.

A prototype has been built to facilitate requirement analysis. The following is a screenshot of the prototype:

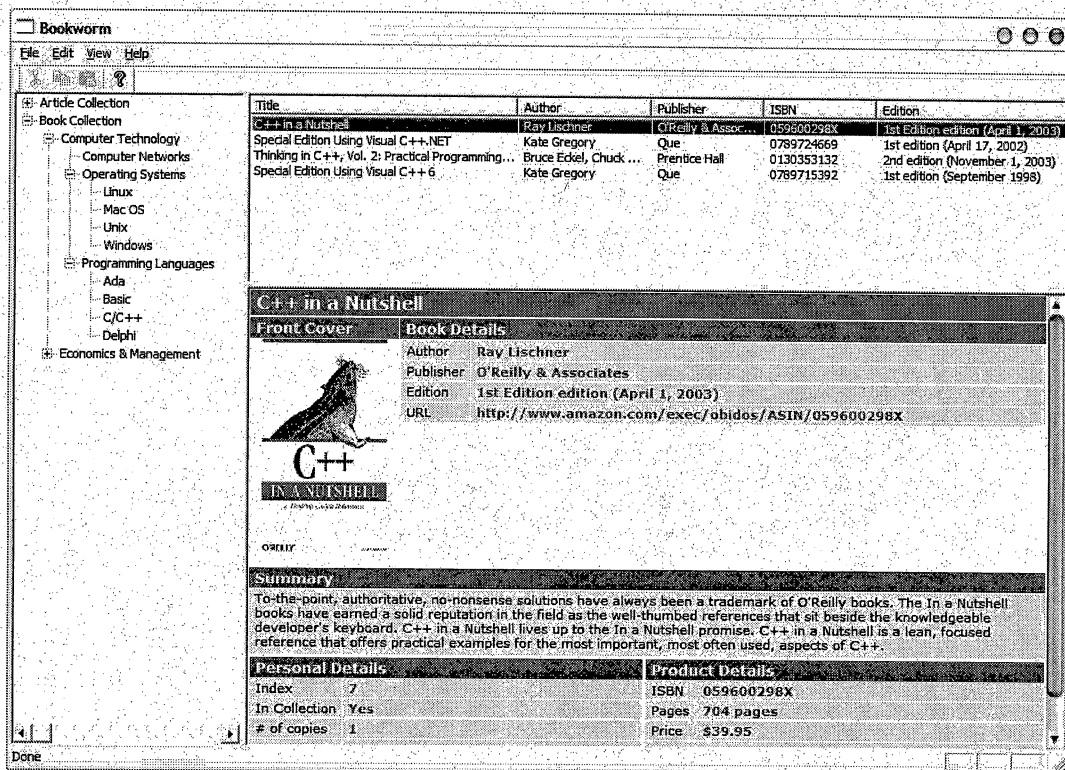


Figure 1. Screenshot of the Prototype

2.2.1.3 Software Interfaces. The major software interface in this system is the interface between the program itself and the backend database. They will communicate using SQL (Standard Query Language) for all database operations.

2.2.1.4 Communication Interfaces. Communication interfaces between this system and remote Internet servers will be implemented with Hypertext Transfer Protocol (HTTP).

2.2.1.5 Memory Constraints. The system does not have any specific memory constraints. It should run adequately

fast on any personal computer that is capable of running the operating system itself.

#### 2.2.2 Product Functions

There are two different roles: normal users and power users. In addition to what a normal user can do, a power user can perform additional tasks which require certain knowledge on computer technology in general. The following is the use case diagram showing the roles and the functions.

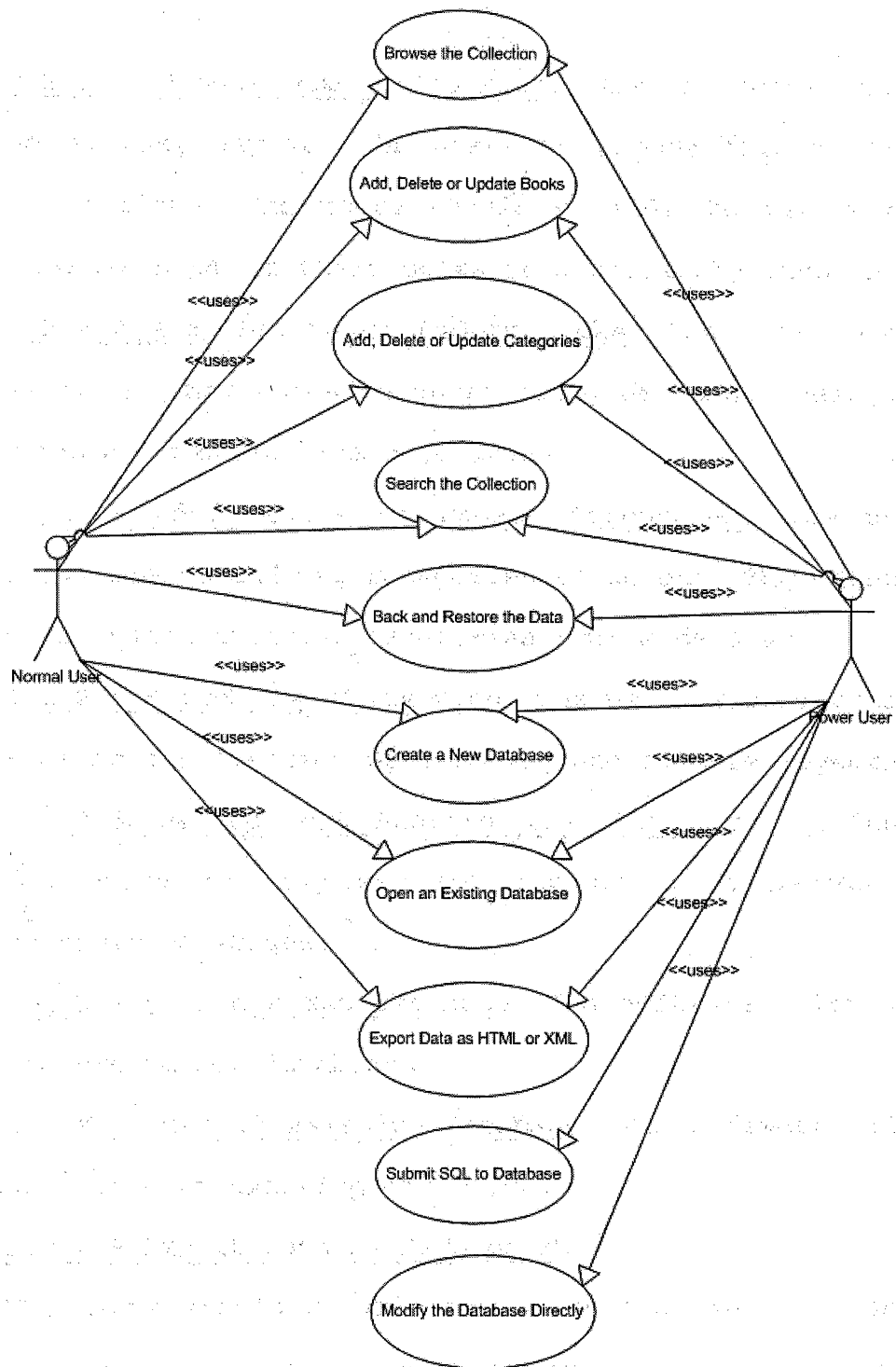


Figure 2. the Use Case Diagram



websites; or export book information as XML so that it can be used to exchange book information with other users.

2.2.2.9 Submit SQL to Database. The software should provide a "Run SQL..." feature so that a user can submit SQL statements to the database, which offers maximum flexibility in manipulating the book information in an eLibrary database. However, it requires the users to be acquainted with SQL and a certain familiarity of eLibrary database schema, so this feature is for advanced users only. The eLibrary database schema should be provided with the software so that the users can refer to it when needed.

2.2.2.10 Modify the Database Directly. If needed, the users should be able to modify the database directly, without using eLibrary at all. It requires the user to be acquainted with database concepts and the eLibrary database schema, so this is also for advanced users only.

### 2.2.3 User Characteristics

There are two different users in the system: normal users and power users. A normal user is expected to have an average knowledge of how to operate a computer and use computer software; no special knowledge is required for normal users. On the other hand, a power user is expected to be familiar with common database concepts, SQL and the eLibrary database schema.

### 2.3 Specific Requirements

The software shall have the following features:

- ◆ Be able to import and export data in XML format so it would be easy to exchange data with other users and other software applications;
- ◆ Export data as HTML and RTF (Rich Text Format) so that the information can be used on websites or in word processors;
- ◆ Be able to create as many subcategories as needed; one item (book or article) can be in multiple categories. For example, a book on computer network security can be in both "Computer network" category and "Information security" category;
- ◆ Be able to backup and restore all the data;
- ◆ Help the user to keep track of the reading progress: which books have been read, or the percentage which has been finished for each individual book;
- ◆ Download book information from the web (such as Amazon.com) by just entering one or more ISBNs, or by a query on book title, author and publisher. Typing from the users will be minimal;
- ◆ Support barcode scanners so that users don't have to type ISBNs manually;

- ◆ Support multi-language so that people from all around the world can choose their preferred interface languages;
- ◆ The view of entries will be fully user configurable, based on XML/XSL or HTML template;
- ◆ Support powerful search capabilities;
- ◆ Records comprehensive loan histories and track overdue books; borrowers' phone numbers and email addresses will be kept in the database in case the user needs to contact the borrowers;
- ◆ Produces detailed and customizable reports for viewing or printing;
- ◆ User-friendly: GUI elements such as drag and drop, context menus and wizards are supported;
- ◆ Utilize the services from other websites if possible, such as NEC Research Institute CiteSeer (<http://citeseer.nj.nec.com> or <http://citeseer.org>).

#### 2.3.1 External Interface Requirements

A graphical user interface shall be provided for the users. After starting the system, the users can then create a new database or open an existing database and do various operations.

### 2.3.2 Functional Requirements

Validity checks should be done on all the inputs. In case of abnormal situations, such as invalid input or overflow, the system should exit gracefully after giving the user an informative message box. Crashes should be avoided.

### 2.3.3 Performance Requirements

The system should run adequately fast on any personal computer that is capable of running the operating system itself.

### 2.3.4 Software System Attributes

2.3.4.1 Security. Since this system is targeted at end users, the system itself does not have security issues.

2.3.4.2 Maintainability. The source code shall be adequately commented; all documents in the software process shall be kept for maintenance purpose. The goal is to make it easy to take over even for somebody who is totally new to this project.

## CHAPTER THREE

### DESIGN AND IMPLEMENTATION

#### 3.1 Design and Implementation Decisions

Before we actually start the design and implementation process, a series of decisions have to be made first. For example, what is the target operating system? How do we store the data? And so on.

##### 3.1.1 Target Operating System

According to a research conducted by IDC, Microsoft Windows Operating systems still dominate the desktop operating system market. Since eLibrary is targeted at end users, obviously it should at least support the Windows operating systems.

Ideally eLibrary is cross-platform, i.e. it would be even better if it can run on different platforms. In that case Java™ seems to be the only viable choice of programming language. However, Java™ programs are compiled into byte code (as opposed to machine code) and they run on top of the Java™ Virtual Machine, which makes them significantly slower. Thus we excluded this possibility.

##### 3.1.2 Data File Format

We can design a new data file format for eLibrary from scratch, or we can use a relational database to store the data. There are two main advantages of using a relational

database: first, it is open and portable. Once the database schema is known, the data contained in the database can be easily used by other applications or ported to other platforms. Second, data integrity can be enforced by a database management system (DBMS). Therefore, we choose to use a relational database to store the book information.

## 3.2 Database Design

### 3.2.1 Database Schema

Database design is an essential part of this project. Based on the analysis, we designed the database using PowerDesigner [7] from Sybase, Inc. PowerDesigner has been one of the leading data modeling tools for years. It supports Conceptual, Logical and Physical Data models, and can check database models for potential errors. The following diagram is the physical data model produced by PowerDesigner:

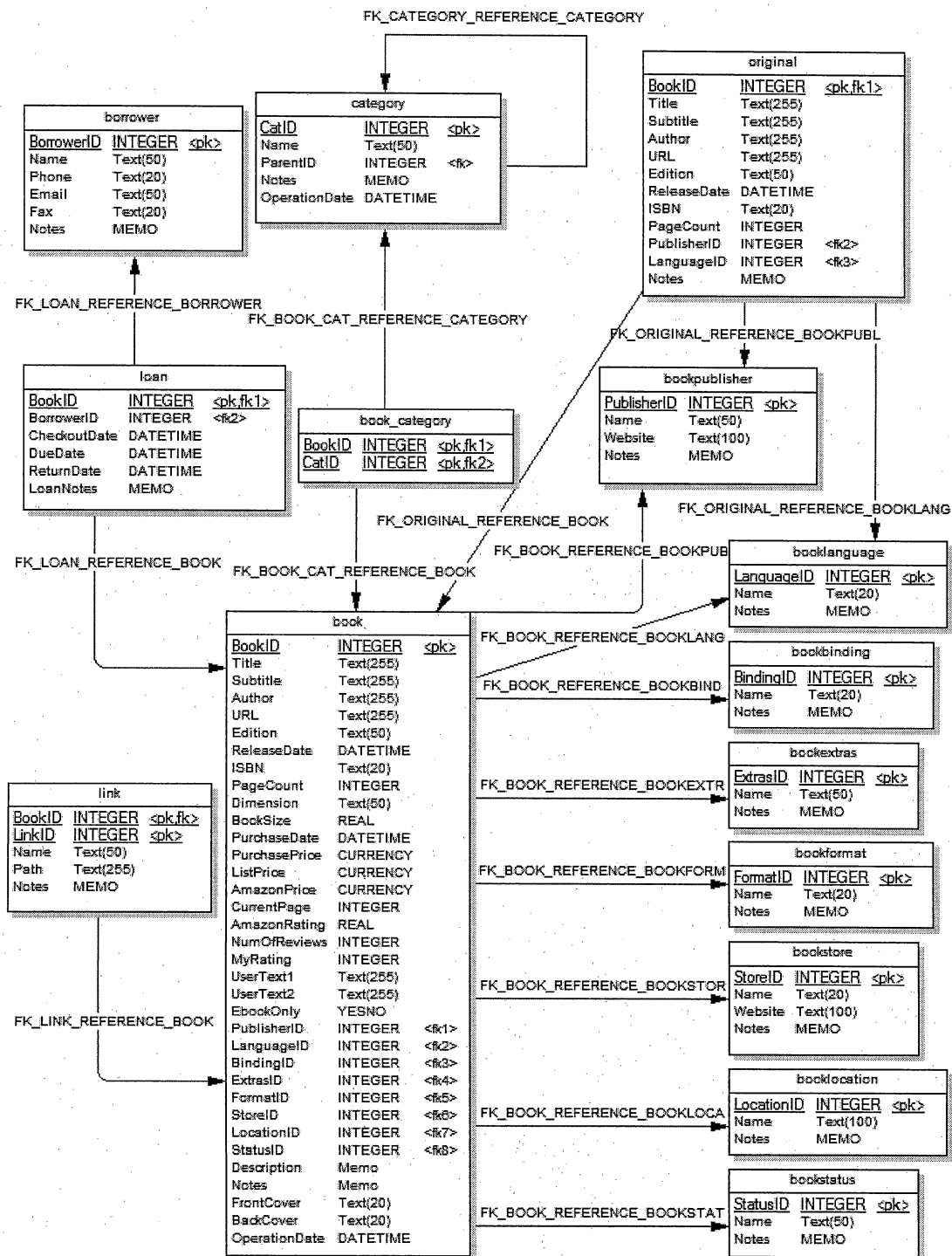


Figure 3. Database Schema

The database schema contains 15 tables, and it is normalized to meet the 3NF.

### 3.2.2 Database Details

Since eLibrary is targeted at the Windows operating system and to be used in a single-user environment, Microsoft Access seems to be the most obvious choice for database management system. Microsoft Access uses the Microsoft Jet Engine. Being fast and efficient, it is the perfect solution for eLibrary to store its data.

3.2.2.1 The "book" Table. This table stores information about a particular book. It has 35 fields, as follows:

BookID: primary key. It denotes the ID of a book.

Title: book's main title.

Subtitle: book's sub-title.

Author: store the authors.

URL: book's URL on the Internet.

Edition: book's edition information.

ReleaseDate: release date.

ISBN: International Standard Book Number.

PageCount: Number of pages.

Dimension: book's dimension information.

BookSize: size for ebooks (in megabytes).

PurchaseDate: purchase date.

PurchasePrice: purchase price.



ListPrice: list price.

AmazonPrice: price on Amazon.com.

CurrentPage: to help the user to keep track of his or her reading progress.

AmazonRating: rating on Amazon.com.

NumOfReviews: number of reviews on Amazon.com.

MyRating: the user's own rating.

UserText1: user-defined text field.

UserText2: user-defined text field.

EbookOnly: denotes if the user only has the book's ebook (no print book in collection).

PublisherID: foreign key. Publisher's ID.

LanguageID: foreign key. ID of the book's language.

BindingID: foreign key. ID of the book's binding.

ExtrasID: foreign key. ID of the book's extras.

FormatID: foreign key. ID of the book's ebook format (such as PDF, CHM etc).

StoreID: foreign key. ID of the store where the book is purchased.

LocationID: foreign key. ID of the location where the book is stored.

StatusID: foreign key. ID of the book's status (such as "In collection", "Wish list" etc).

Description: book's description or abstract.

Notes: user's notes about this book.

FrontCover: the file name of the book's front cover picture.

BackCover: the file name of the book's back cover picture.

OperationDate: when this book's information is last updated.

3.2.2.2 The "original" Table. This table stores the original information about a particular book, in case the book is translated from another language. It has 12 fields, as follows:

BookID: primary key. It denotes the ID of a book.

Title: book's main title.

Subtitle: book's sub-title.

Author: store the authors.

URL: book's URL on the Internet.

Edition: book's edition information.

ReleaseDate: release date.

ISBN: International Standard Book Number.

PageCount: Number of pages.

Dimension: book's dimension information.

PublisherID: foreign key. Publisher's ID.

LanguageID: foreign key. ID of the book's language.

Notes: user's notes about this book.

3.2.2.3 The "link" Table. This table stores the "related links" of a book. A "related link" can be the author's website or any other related URLs, or an ebook file or folder located on the user's hard drive. Unlimited number of links can be added for one book.

The "link" table has 5 fields, as follows:

BookID: primary key. The ID of the book.

LinkID: primary key. The ID of the link.

Name: name of the link, such as "author's website".

Path: path of the link. For example, "http://www.BruceEckel.com" or "C:\ebooks\mybook.pdf".

Notes: user's notes about this link.

3.2.2.4 The "category" Table. Users can define their own category and the category information is stored in this table. It has the following 5 fields:

CatID: primary key. The ID of the category.

Name: name of the category.

ParentID: the ID of this category's parent category.

For the root category, this field is NULL.

Notes: user's notes about this category.

OperationDate: when this category's information is last updated.

3.2.2.5 The "book category" Table. This table acts as a bridge between the "book" table and the "category" table.

Its sole purpose is to turn an m:n relationship into two 1:n relationships. It is this table that allows us to store one book in multiple categories. For example, a book on "Network Security" can be in both "Computer Networks" category and "Information Security" category.

The "book\_category" table only has two fields:

BookID: primary key. The ID of the book.

CatID: primary key. The ID of the category.

3.2.2.6 The "borrower" Table. The "borrower" table is to keep borrowers' information. It has the following 6 fields:

BorrowerID: primary key. The ID of the borrower.

Name: borrower's name.

Phone: borrower's phone number.

Email: borrower's email address.

Fax: borrower's fax number.

Notes: user's notes about this borrower.

3.2.2.7 The "loan" Table. This table is used to keep track of book loans. It has 6 fields, as follows:

BookID: primary key. The ID of the book.

BorrowerID: primary key. The ID of the borrower.

CheckoutDate: checkout date.

DueDate: due date.

ReturnDate: the actual return date.

LoanNotes: user's notes about this loan.

3.2.2.8 The lookup Tables. There are eight lookup tables in the database. Their schema consists of an Identification (ID) field, a "name" field and a "notes" field. The "name" field defines the value of the ID, and the "notes" field can keep a user's notes for a particular item. For some table, there is an extra field called "website". For example, in the "bookpublisher" table, the "website" field records the publisher's website.

These lookup tables are populated with pre-defined data when the system is setup. Additional records can be added later.

The eight lookup tables in the eLibrary database are as follows:

Bookpublisher: publisher information.

Booklanguage: language information.

Bookbinding: binding information, such as "Hardcover" or "Paperback".

Bookextras: some books have extras, such as a "CD-ROM" or "Floppy Disk".

Bookformat: file format of ebook files, such as "CHM" or "PDF".

Bookstore: stores where the book is purchased. For example, "Amazon.com" or "Borders".

Booklocation: locations where the book is stored, such as "My big book shelf".

Bookstatus: book status, such as "In collection" or "For sale".

### 3.3 Program Design and Implementation

#### 3.3.1 System Architecture

Information retrieval and information display are two fundamental components of eLibrary.

3.3.1.1 Information Retrieval. Amazon.com's web services [1] offer software developers the opportunity to integrate Amazon.com features and content directly into other websites or desktop applications using either SOAP or XML over HTTP. This project will use XML over HTTP to retrieve book information (such as book title, author, publisher, price etc) from Amazon so that the users don't have to type them manually. eLibrary will query online by one or more ISBNs, or by a combination of book title, author and/or publisher. The architecture is shown below.

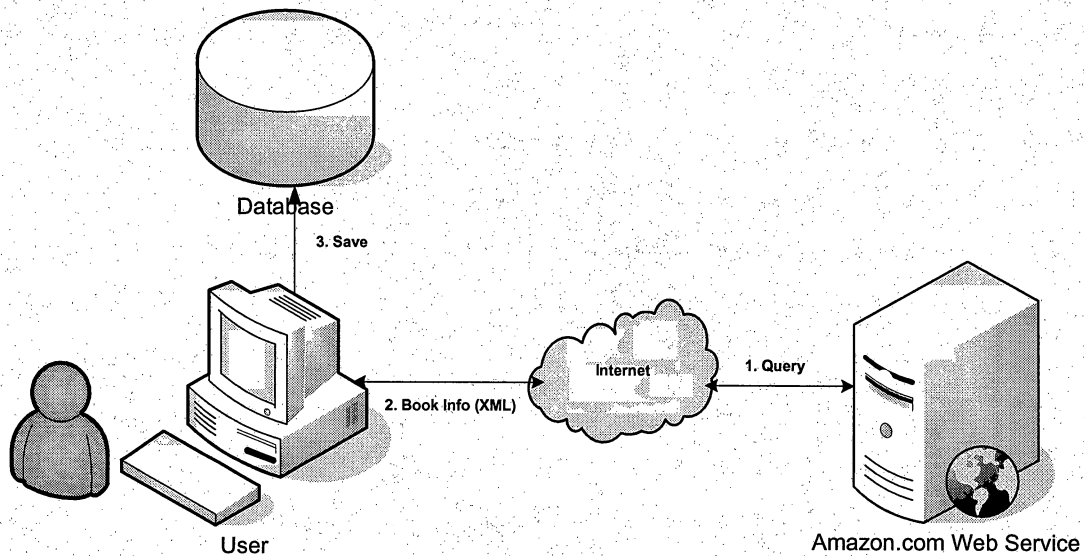


Figure 4. Information Retrieval Process

Three steps are involved in the information retrieval process:

1. eLibrary sends the user's query term (such as an ISBN) to Amazon.com web service;
2. eLibrary receives the response from Amazon.com. The data is in XML format; eLibrary parses the XML received and display the data;
3. The user makes necessary modifications to the data and enters additional information, and eLibrary saves the data to a local database.

eLibrary uses Microsoft XML Core Services (MSXML) [6] to parse the XML data returned from Amazon.com. MSXML is an XML parser. It allows developers to build high-performance

XML-based applications that provide a high degree of interoperability with other applications that adhere to the XML 1.0 standard.

3.3.1.2 Information Display. When the user browses through his or her book collection, book information is displayed as HTML. There are also three steps involved in the information display process, as shown by the following diagram:



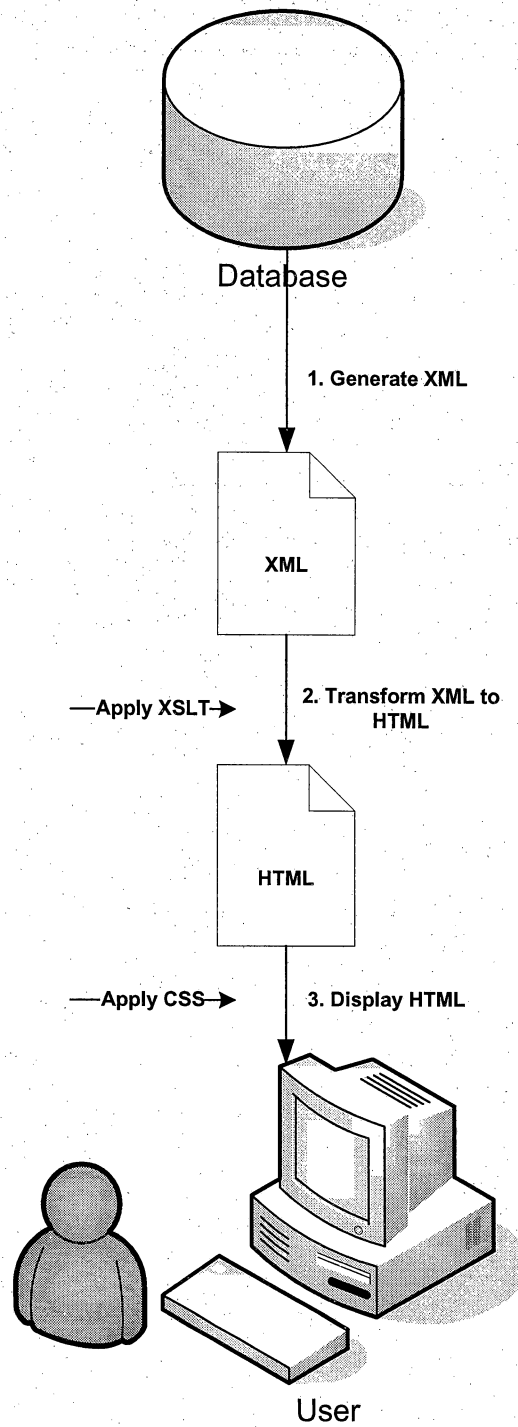


Figure 5. Information Display Process

1. The book information is retrieved from the local database, and an XML string containing the book information is generated;
2. The XML string is transformed into an HTML string by applying an external XSLT style sheet;
3. The HTML string is displayed on the user's screen, after applying an external CSS style sheet.

eLibrary uses Microsoft ActiveX Data Objects (ADO) [4] as a data access method to retrieve information from the database. ADO enables the application to access and manipulate data from a variety of sources through an OLE DB provider. Its primary benefits are ease of use, high speed, low memory overhead, and a small disk footprint.

### 3.3.2 Program Details

Microsoft Visual C++ .Net 2003 [5] with MFC (Microsoft Foundation Class) [3] is used as the primary development tool in developing eLibrary. Visual C++ has been one of the most powerful tools for developing Windows-based applications. It offers many features that make writing and maintaining complex code easier and safer. eLibrary is a standard Windows program. Just like many other Windows

programs, it has a menu bar, a tool bar, a client area, and a status bar, as seen in the following figure.

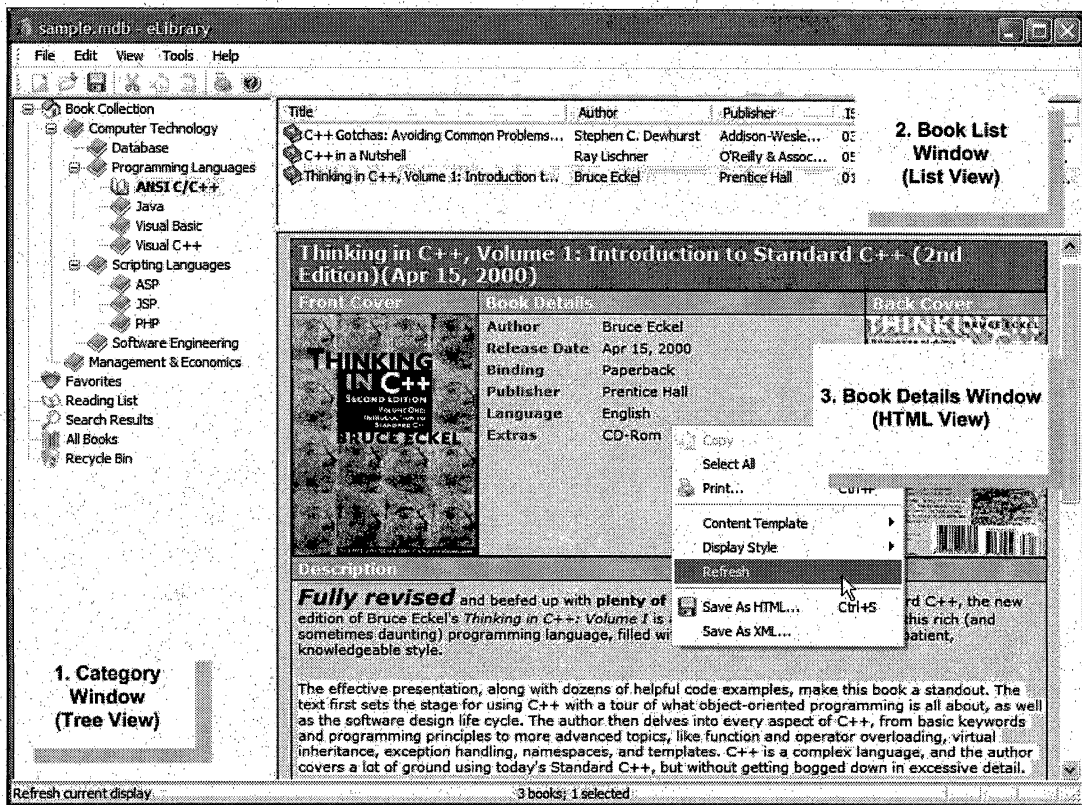


Figure 6. eLibrary Screenshot

The client area is divided into three panes: the left side Category Window, the top-right Book List Window and the bottom-right Book Details Window. There is a separate context menu associated with each window.

**3.3.2.1 The Category Window.** The category window uses a tree structure to display all the categories in a database.

When the user clicks a category in the Book Category window, the Book List window displays a list of books in the current selected category. eLibrary defines two kinds of categories: normal categories and system categories. The user can add, delete or update a normal category, but cannot modify system categories. Currently there are six system categories:

1. Root category (CatID = 1): The root of all categories.
2. Favorites (CatID = 2): Contains the user's favorite books.
3. Search Results (CatID = 3): When the user performs a search operation, the search results (a list of books) will be put in this category.
4. Reading List (CatID = 4): The user can add books that he or she is currently reading to this category so that those books could be easily found.
5. All Books (CatID = 5): When the user clicks this category in the Book Category window, the Book List window will display all the books in the database.
6. Recycle Bin (CatID = 6): Contains deleted books. The user can still recover deleted

books from the Recycle Bin by Drag & Drop or clipboard operations (Copy & Paste).

The C++ class associated with the Book Category window is `CeLibTreeView`, which is inherited from `CTreeView`. `CTreeView` wraps a "tree view control". MFC's `CTreeView` class enables programmers to create views similar to the one featured in the left pane of Windows Explorer. Tree views display treelike structures containing items composed of text and images. Items can have subitems, and collections of subitems, or subtrees, and can be expanded and collapsed to display and hide the information contained therein. Tree views are ideal for depicting data that's inherently hierarchical, such as the directory structure of a hard disk.

Figure 7 shows the class diagram of `CeLibTreeView`.

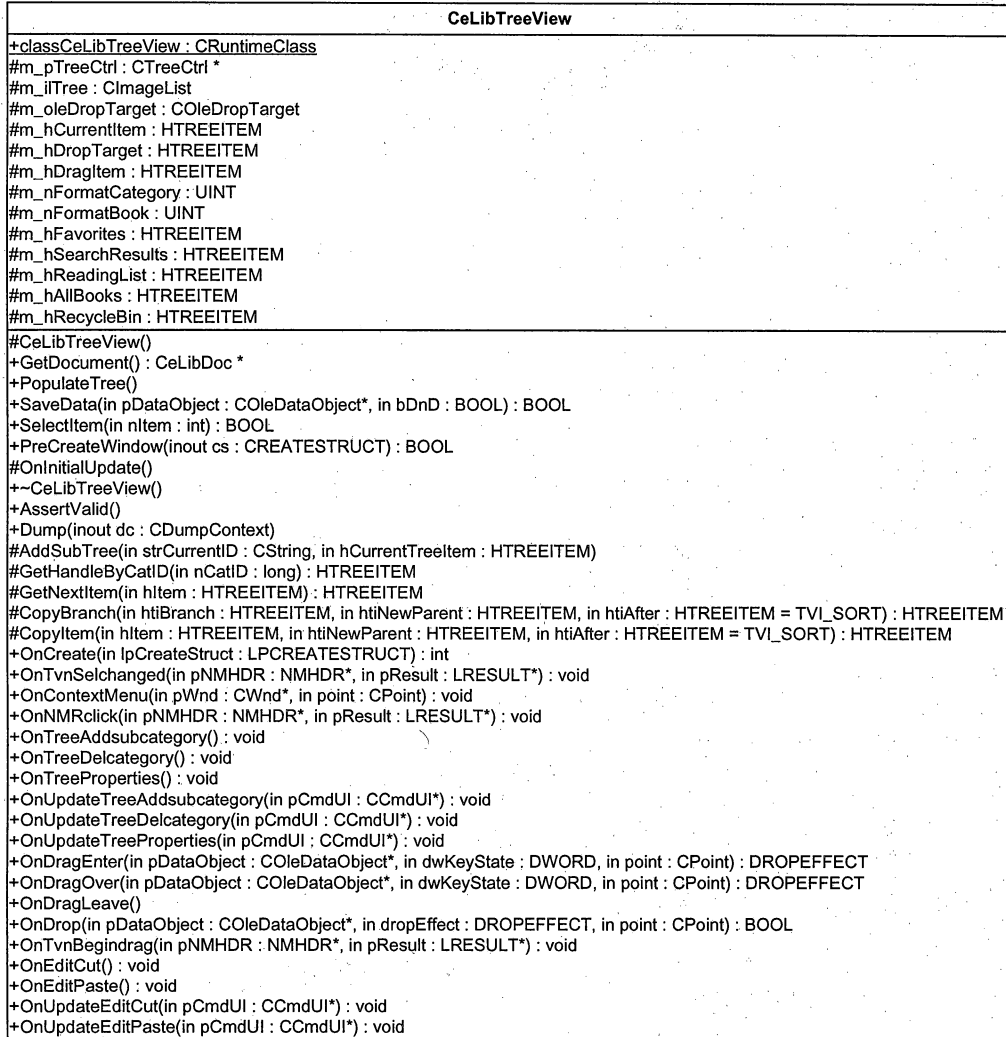


Figure 7. Class Diagram of CellibTreeView

3.3.2.2 The Book List Window. The Book List window displays a list of books in a particular category. When the user selects one or more items in the Book List window, it sends a message to the Book Details window and the Book

Details window will refresh itself so that the user can see detailed information about the selected book(s).

The C++ class associated with the Book List window is `CeLibListView`, which is inherited from `CListView`. The `CListView` class encapsulates the functionality of a "list view control", which displays a collection of items each consisting of an icon (from an image list) and a label. List views are similar to tree views in that they provide a powerful infrastructure for presenting complex collections of data to the user. But whereas tree views are ideal for depicting hierarchical relationships, list views are best suited for presenting "flat" collections of data, such as lists of file names. Like items in a tree view, items in a list view can include both text and images. In addition, items can have text-only subitems containing additional information about the associated items. The subitems are visible when the control is in "report" mode, which is one of four presentation styles that a list view supports. The other presentation styles are large icon mode, small icon mode, and list mode. `eLibrary` mainly uses the "report" mode.

Figure 8 shows the class diagram of `CeLibListView`.

CeLibListView
<pre> +class CeLibListView : CRuntimeClass #m_pListCtrl : CListCtrl * #m_iList : CImageList #bRefreshHtmlNeeded : BOOL #m_SortInfo : SortInfo #m_ColSortGUI : CHeaderColSortMark  #CeLibListView() #~CeLibListView() #PreCreateWindow(inout cs : CREATESTRUCT) : BOOL #RefreshBookIDArray() #DeleteSelectedBooks() #CompareListItems(in IParam1 : LPARAM, in IParam2 : LPARAM, in IParamSort : LPARAM) : int +GetDocument() : CeLibDoc * +AssertValid() +Dump(inout dc : CDumpContext) +Refresh() +OnCreate(in lpCreateStruct : LPCREATESTRUCT) : int +OnTimer(in nIDEvent : UINT) : void +OnLvnColumnClick(in pNMHDR : NMHDR*, in pResult : LRESULT*) : void +OnContextMenu(in pWnd : CWnd*, in point : CPoint) : void +OnLvnItemchanged(in pNMHDR : NMHDR*, in pResult : LRESULT*) : void +OnListAddnewbook() : void +OnListDeletebook() : void +OnListProperties() : void +OnEditSelectAll() : void +OnUpdateListAddnewbook(in pCmdUI : CCmdUI*) : void +OnUpdateListDeletebook(in pCmdUI : CCmdUI*) : void +OnUpdateListProperties(in pCmdUI : CCmdUI*) : void +OnUpdateEditSelectAll(in pCmdUI : CCmdUI*) : void +OnNMDblclk(in pNMHDR : NMHDR*, in pResult : LRESULT*) : void +OnEditCut() : void +OnUpdateEditCut(in pCmdUI : CCmdUI*) : void +OnLvnBeginDrag(in pNMHDR : NMHDR*, in pResult : LRESULT*) : void +OnEditCopy() : void +OnUpdateEditCopy(in pCmdUI : CCmdUI*) : void +OnEditPaste() : void +OnUpdateEditPaste(in pCmdUI : CCmdUI*) : void +OnSendtoFavorites() : void +OnSendtoReadinglist() : void +OnSendtoSearchresults() : void +OnUpdateSendtoFavorites(in pCmdUI : CCmdUI*) : void +OnUpdateSendtoReadinglist(in pCmdUI : CCmdUI*) : void +OnUpdateSendtoSearchresults(in pCmdUI : CCmdUI*) : void </pre>

Figure 8. Class Diagram of CeLibListView

3.3.2.3 The Book Details Window. The Book Details window displays detailed information of the selected book(s).



It uses an external XSL file to control its content (what to display, what not to display, etc). In eLibrary interface, an XSL file is called a "content template", which resides in the "Styles" directory. eLibrary supports up to 100 different content templates, and the user can switch content template at any time.

The Book Details window also uses an external CSS file to control its display style (font, color etc). The CSS files should be put in the "Styles/CSS" directory. Again, eLibrary supports up to 100 different display styles, and the user can switch display style at any time.

The C++ class associated with the Book Details window is CeLibHtmlView, which is inherited from CHtmlView. CHtmlView is one of MFC's most powerful new classes, which converts the WebBrowser control that's the heart and soul of Microsoft Internet Explorer into a full-fledged MFC view. CHtmlView displays HTML documents. The developer provides a URL, which can reference a document on the Internet, on an intranet, or even on a local hard disk, and CHtmlView displays the document the same way Internet Explorer displays it. CHtmlView is also an Active Document container, which means it can be used to display documents created by Microsoft Word, Microsoft Excel, and other Active Document servers. It can even display the contents of folders on a

hard disk – just like Internet Explorer. Since CHtmlView derives most of its functionality from the WebBrowser control, and because the WebBrowser control is part of Internet Explorer, an application that uses CHtmlView can be run only on systems equipped with Internet Explorer 4.0 or later.

The following diagram is the class diagram of CeLibHtmlView.

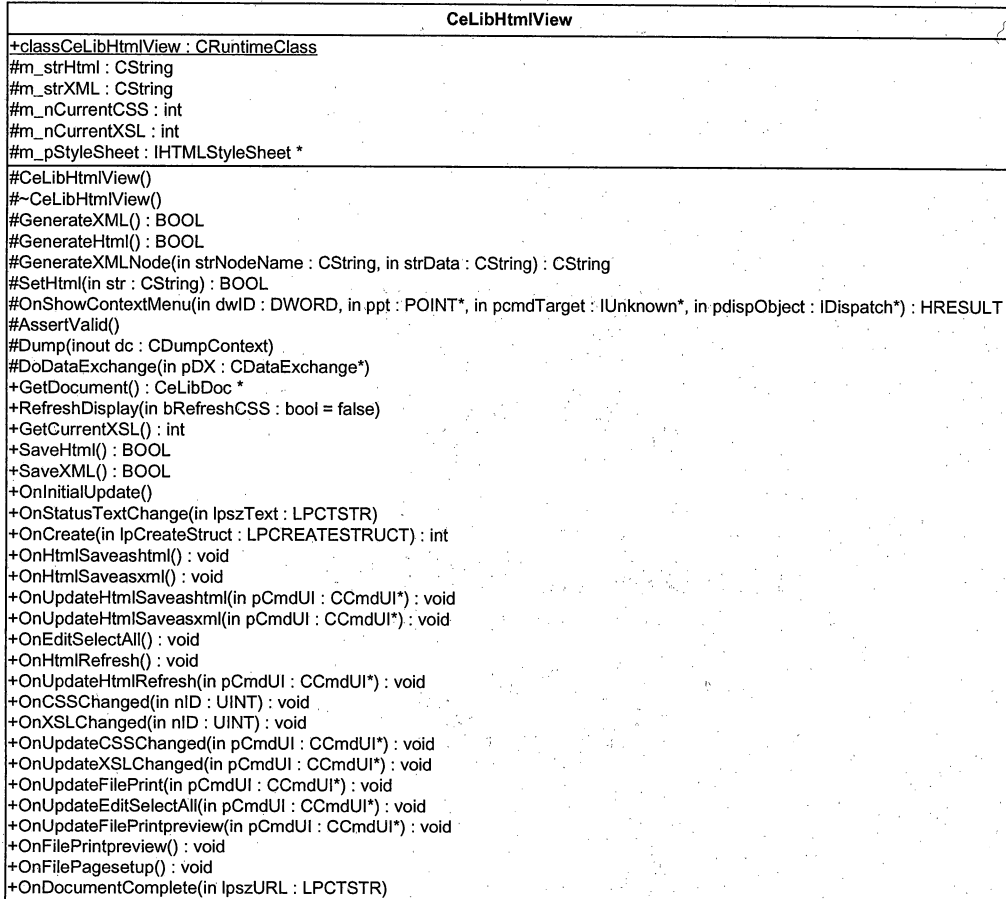


Figure 9. Class Diagram of CeLibHtmlView

## CHAPTER FOUR

### DEPLOYMENT

#### 4.1 System Requirements

eLibrary has no special requirements on hardware. It should run adequately fast on any personal computer that is capable of running the Windows operating system itself.

On the software side, eLibrary is able to run on all major win32 platforms, from Windows 98 to the latest Windows XP and Windows 2003. However, since eLibrary uses MSXML to handle XML parsing, Internet Explorer 5.0 or higher should be installed on the destination computer. Nowadays most of the personal computers running Windows operating systems have Internet Explorer 6.0 installed, so it is safe to say that this is a reasonable requirement.

#### 4.2 Installation

##### 4.2.1 Installer

An installer is the first experience of a user with the application. Slow or unsuccessful software installations are among the most irritating computer problems. A quick and user friendly installer is therefore an essential part of the software product.

eLibrary uses NSIS to create an installer. NSIS (Nullsoft Scriptable Install System) is a tool that allows developers to create professional installers for Windows. It is released under an open source license and is completely free for any use.

NSIS can create Windows installers that are capable of installing, uninstalling, setting system settings, extracting files, etc. Because NSIS is based on script files, developers can use it to create both simple and advanced installers. However, NSIS is only a "script compiler"; it does not come with any script/dialog editors. HM NIS Edit is a free visual environment for NSIS. It makes creating and maintaining NSIS install scripts quick and easy. So we use HM NIS Edit to generate an install script; then use NSIS to compile the install script and produce our installer.

#### 4.2.2 Installation of eLibrary

With the eLibrary installer, it is very easy to install eLibrary. Just execute the installer and follow the instructions. All the necessary files will be extracted from the installer and copied to the specified directory.

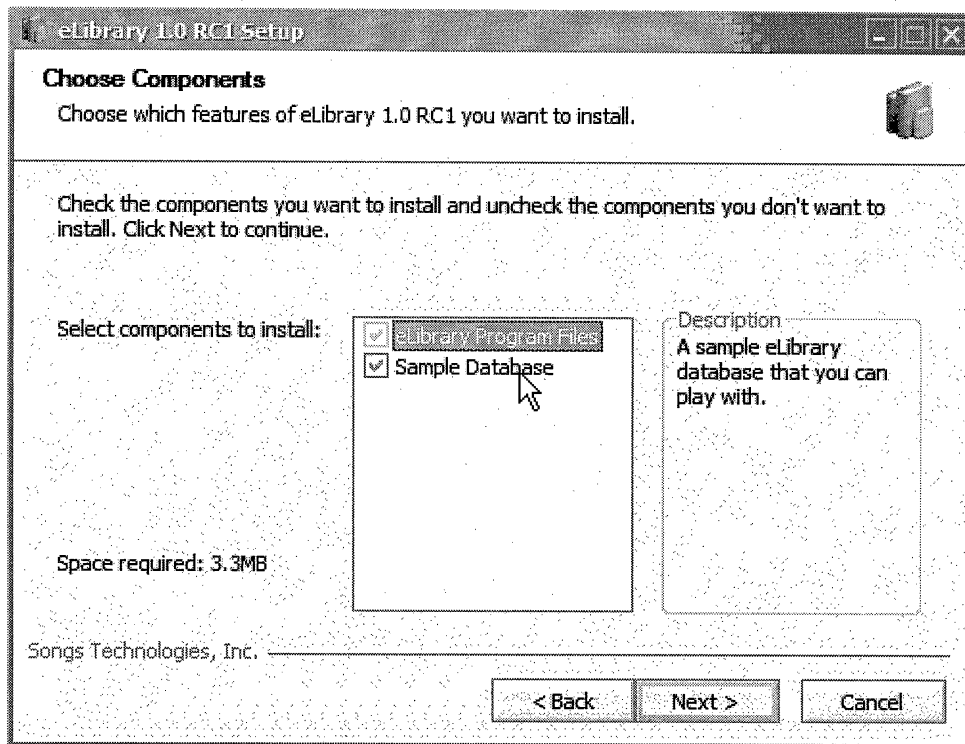


Figure 10. eLibrary Installer

## CHAPTER FIVE

### CONCLUSION AND FUTURE DIRECTIONS

#### 5.1 Conclusion

eLibrary is a book management software application that runs on Microsoft Windows platforms. It uses a relational database to store the book information; the data contained in the database can be easily used by other applications. Using tree structure to manage book categories, eLibrary has a familiar Windows Explorer-like user interface. It can download book information from the Internet automatically; the user only need to type the ISBN or simply uses his or her barcode scanner. eLibrary uses XML/XSL to display book details; the content template and display style are completely configurable by the user. eLibrary is a complete solution for people who wish to build their own personal electronic libraries.

Six beta versions have been released to the public, and the first non-beta version (V1.0 RC1) of eLibrary was released on April 23, 2004. It has gained much popularity among the users since its initial release. Thousands of people have downloaded eLibrary and many of them are using it on a regular basis. In addition, it has been submitted to some software download sites and many sites have given it the highest five-star award.

## 5.2 Future Directions

Due to time constraints, some features are not implemented yet and will be implemented in the near future.

Loan management: to keep track of book loans. The database already has the corresponding tables reserved for loan management, but the functionality is not implemented in the program yet.

Plug-in support: plug-ins are optional software additions that enhance and/or add functionality to the main software application. With plug-in support, capable users and developers can develop their own eLibrary plug-ins, and the whole user community can benefit from that. (eLibrary will communicate with plug-ins via XML. For example, a plug-in that imports BibTex will read a BibTex file and generate an eLibrary XML stream based on the file content. When eLibrary receives the XML stream, it saves the information to database. The eLibrary XML definition is available online at <http://forum.songstech.com/viewthread.php?tid=35>.)

Help file and website improvements: the help file is still not very complete; also, the eLibrary homepage (<http://songstech.com> or <http://www.eLibPro.com>) need to be re-designed so that it can provide more relevant information for the users.



APPENDIX  
LIST OF SOURCE CODE FILES

The following is a list of source code files, in alphabetical order:

AddBookDlg.cpp

AddBookDlg.h

AlphaImageList.cpp

AlphaImageList.h

AlphaToolBar.cpp

AlphaToolBar.h

BasicSearchDlg.cpp

BasicSearchDlg.h

BatchQueryDlg.cpp

BatchQueryDlg.h

BookBasicPage.cpp

BookBasicPage.h

BookCoversPage.cpp

BookCoversPage.h

BookDescriptionPage.cpp

BookDescriptionPage.h

Bookinfo.cpp

Bookinfo.h

BookLink.h

BookLinksPage.cpp

BookLinksPage.h

BookNotesPage.cpp

BookNotesPage.h  
BookPersonalPage.cpp  
BookPersonalPage.h  
BookProperties.cpp  
BookProperties.h  
CategoryGeneralPage.cpp  
CategoryGeneralPage.h  
CategoryProperties.cpp  
CategoryProperties.h  
eLib.cpp  
eLib.h  
eLibDoc.cpp  
eLibDoc.h  
eLibHtmlEditor.cpp  
eLibHtmlEditor.h  
eLibHtmlView.cpp  
eLibHtmlView.h  
eLibListView.cpp  
eLibListView.h  
eLibTreeView.cpp  
eLibTreeView.h  
explorer.cpp  
explorer.h  
HeaderColSortMark.cpp

HeaderColSortMark.h  
HyperLink.cpp  
HyperLink.h  
InputSQLDlg.cpp  
InputSQLDlg.h  
LinkGeneralPage.cpp  
LinkGeneralPage.h  
LinkProperties.cpp  
LinkProperties.h  
MainFrm.cpp  
MainFrm.h  
MenuBar.cpp  
MenuBar.h  
OnlineQueryResultsDlg.cpp  
OnlineQueryResultsDlg.h  
OpenDlgEx.cpp  
OpenDlgEx.h  
Options.cpp  
Options.h  
OptionsDefaultPage.cpp  
OptionsDefaultPage.h  
OptionsUIPage.cpp  
OptionsUIPage.h  
PictureEx.cpp

PictureEx.h

Registry.cpp

Registry.h

Resource.h

stdafx.cpp

stdafx.h

utils.cpp

utils.h

## BIBLIOGRAPHY

- [1] Amazon Web Services. <http://www.amazon.com/webservices>
- [2] Bruce Eckel's MindView, Inc. <http://www.BruceEckel.com/>
- [3] Jeff Prosise. *Programming Windows With MFC*, 2nd ed. Microsoft Press, 1999.
- [4] Microsoft ActiveX Data Objects. <http://www.microsoft.com/ado>
- [5] Microsoft Visual C++. <http://msdn.microsoft.com/visualc>
- [6] Microsoft XML SDK. <http://www.microsoft.com/xml>
- [7] Sybase PowerDesigner. <http://sybase.com/powerdesigner>