A Web Database Development Course and a Unique Problem-Solving Project

Charles R. Moen, M.S.[1] and Morris M. Liaw, Ph.D.[2]

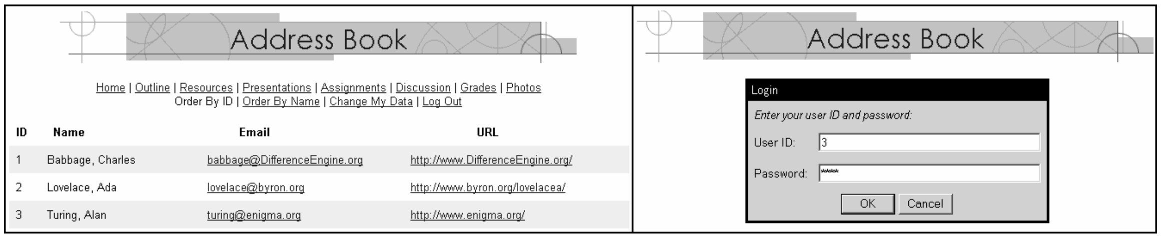
Abstract

Today, the content of most Web sites contains dynamically generated pages that present information stored in a database, and computer science students are eager to learn how to develop these database-driven Web sites. This paper presents how students at the University of Houston - Clear Lake (UHCL) learn about Web database development in a course that emphasizes project-building activities, and it looks at the implementation details of a successful student project, the Online Address Book. Four unique problems and the solutions that were implemented in this project are explained: 1) editing multiple students' data; 2) checking the data with a regular expression; 3) stale session variables; and 4) potential security violations.

The Online Address Book

At UHCL, students have the opportunity to learn how to develop database-driven Web sites by taking CSCI 5633 Web Database Development. [1] In this course, the emphasis is on developing practical skills, and its most important feature is a semester-long project that produces a fully functional Web site that uses a backend database. This section looks at how a successful student project, the Online Address Book was implemented.

The Online Address Book project is a database-driven Web application that displays the names, email addresses, and home page addresses of all of the students in the class. Other requirements are: 1) the user must be able to display the names sorted either by the student's last name or by their user ID number; 2) each student in the Address Book must be able to edit his or her own name and address information; 3) password authentication is required before a user can edit the data; and 4) the data must be maintainable by a user with an administrator role.


Address Book
Home | Outline | Resources | Presentations | Assignments | Discussion | Grades | Photos
Order By ID | Order By Name | Change My Data | Log Out
Address Book
Login
OK
Cancel

ID	Name	Email	URL
1	Babbage, Charles	babbage@DifferenceEngine.org	http://www.DifferenceEngine.org/
2	Lovelace, Ada	lovelace@byron.org	http://www.byron.org/lovelacea/
3	Turing, Alan	turing@enigma.org	http://www.enigma.org/

Enter your user ID and password:

User ID: 3

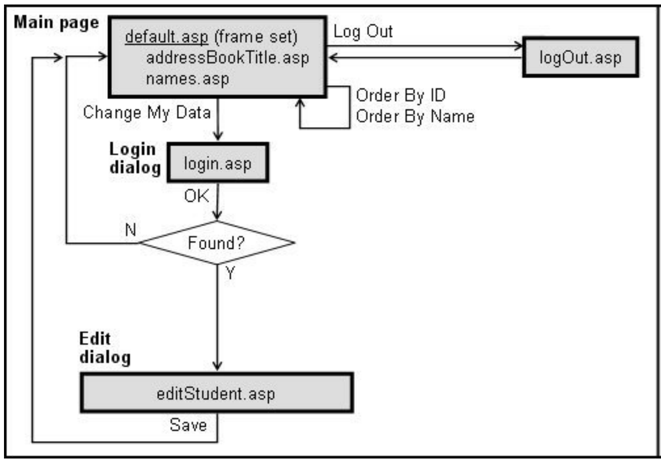
Password: ****

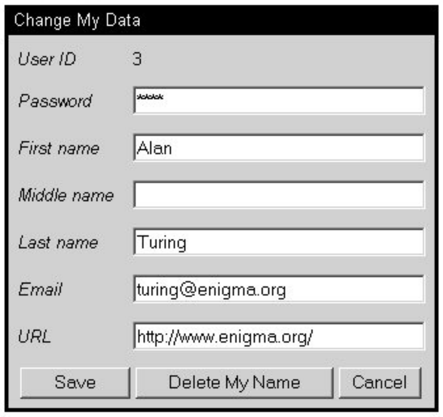
Fig. 1. Left, main page of the Online Address Book. Right, login dialog

The user interface for the Address Book is shown in Fig. 1 on the left. This view is the main page as seen by any user who navigates to the Web site for the course, and it is created dynamically by server-side scripts on the server. The scripts receive the course ID in the query string of the HTTP request. Then the course ID is used in a SQL query to retrieve the records of all the students in that course, and these records are added to the table displayed on the page. The default view shown in the figure displays the records in the order of their user ID number. Alternatively, they can be displayed in the alphabetic order of the students' last names by clicking on the link labeled "Order By Name."

If any of the students in the class wish to make changes to their own data, they can click on the link, "Change My Data," which brings up the login dialog shown in Fig. 1 on the right.

After logging in successfully, the student can either change his or her own data, or delete it from the database. The edit dialog that the student uses to make these changes is shown in Fig. 2 on the right side, and the flow chart showing the sequence of these actions is shown on the left side of the figure.



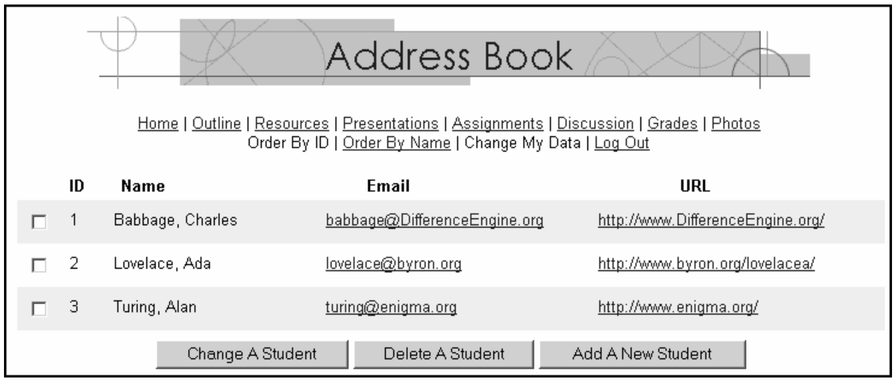

Change My Data

User ID	3
Password	****
First name	Alan
Middle name	
Last name	Turing
Email	turing@enigma.org
URL	http://www.enigma.org/

Fig. 2. Left, flow chart showing actions available to student users. Right, edit dialog.

Additional Features for the Teacher

There are extra features for teachers, but to gain access to this additional functionality, the user has to log in as an administrator. To do so, the user must click on "Change My Data" and enter the appropriate user ID and password. The login script validates whether the user is an administrator by examining a Boolean "Admin" field in that user's database record. Once the teacher is logged in, a "teacher" session variable is set to "true" so that the pages visited after the initial login can recognize the user's status.


Address Book
Home | Outline | Resources | Presentations | Assignments | Discussion | Grades | Photos
Order By ID | Order By Name | Change My Data | Log Out

	ID	Name	Email	URL
☐	1	Babbage, Charles	babbage@DifferenceEngine.org	http://www.DifferenceEngine.org/
☐	2	Lovelace, Ada	lovelace@byron.org	http://www.byron.org/lovelacea/
☐	3	Turing, Alan	turing@enigma.org	http://www.enigma.org/

Change A Student
Delete A Student
Add A New Student

Fig. 3. The page seen by a teacher who is logged in and ready to change the data

After logging in successfully, the teacher will see the Web page shown in Fig. 3. This view shows all the students in the class, and the teacher will be able to add, delete, and edit the data for any student displayed here by selecting the records to change and then clicking the appropriate button at the bottom of the page.

Solutions for Some Special Problems

Two of the solutions developed for the Address Book are unique, and this section examines them in detail.

```
set recordsToUpdate = server.createObject("scripting.dictionary")
count = 0

'Each checkbox's name is the number for the student's user ID
'Get the numbers of all the checkboxes that were checked
for each fld in request.form
      if isNumeric(fld) then
            count = count + 1
            recordsToUpdate(count) = fld
      end if
next

if request("btnChange") <> "" then
      set session.contents("changes") = recordsToUpdate
      session.contents("changeCount") = count
```

Fig. 4. Code within updateStudent.asp that collects the IDs of students to be edited

1. Editing Multiple Students

One special problem was the need for the teacher to have a convenient way to edit multiple students' data. To solve this problem, checkboxes were added, shown in Fig. 3, so that the teacher can select multiple students to change by clicking their checkboxes.

This feature was implemented by using session variables and scripts to collect IDs of the records marked for changes and send them one-by-one to the "edit dialog" page.

The procedure starts when the teacher clicks the checkboxes of the students to be changed and clicks on "Change A Student." At that point, an HTTP post request containing all of the selected checkbox elements is submitted to updateStudent.asp. The name attribute of each of these checkboxes is the user ID of the student that it represents. The code in the script updateStudent.asp collects these user IDs from the request and adds them to a dictionary object, "recordsToUpdate," along with a key which is the value of a counter incremented for each student record. After the IDs of all the checked records have been collected in this way, recordsToUpdate is then put in a session variable called "changes," and the final count is put in "changeCount." This code is shown in Fig. 4.

The updateStudent.asp script then sends the request to getMe.asp which puts the user ID of the first student into the query string and decrements "changeCount." Then the request is forwarded to editStudent.asp where the user ID is used in a SQL query to retrieve the first student's data and display it in the edit dialog box.

After making changes to the data in the dialog, the teacher must click on its "Save" button. This button submits the data to the database and then sends a request to getMe.asp again. There the ID of the next student is taken from "changes" and put into the query string, and the request is sent back to editStudent.asp.

This iterative process continues until changeCount is 0, and there are no more user ID numbers left. Fig. 5 shows the steps in the process.

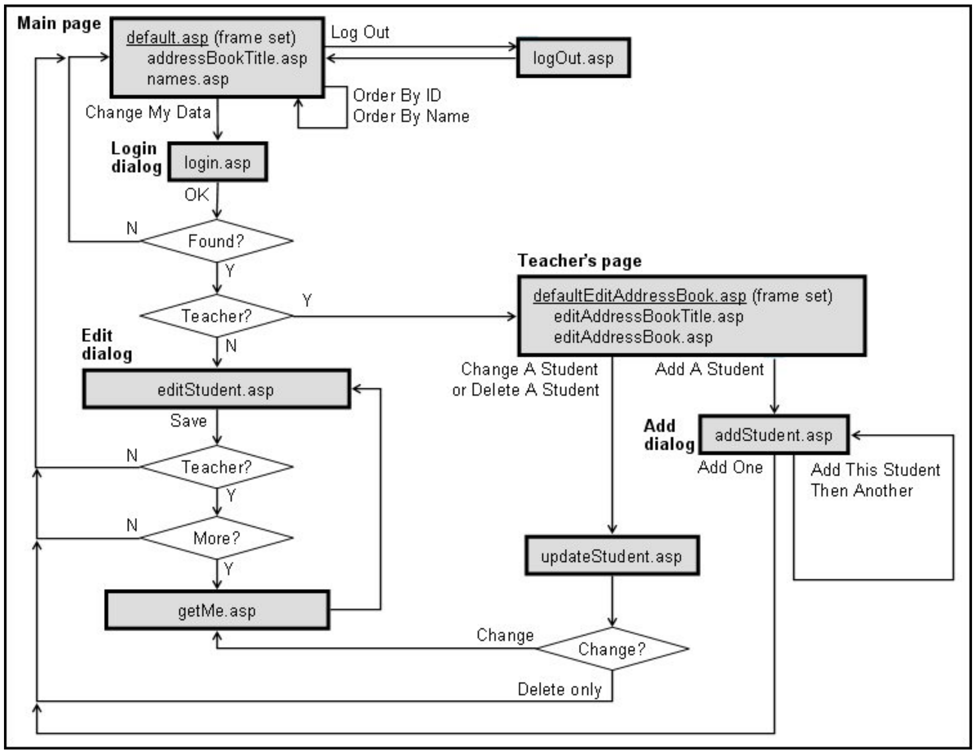


Fig. 5. Flow chart showing actions available to teachers

2. Checking the Data with a Regular Expression

Another problem was caused by inconsistencies in how the users entered their data for the address book. Each entry in the column that displays the URL for the home page of that particular student is designed to be a link to the page. Unfortunately, not all of the students would remember to enter “HTTP://” as part of their home page address, and then the links failed to work.

```
<%
'Did the user put in http in his URL?
thisHome = objRecordset("Homepage")
set objRegExp = new RegExp
objRegExp.pattern = "http"
objRegExp.ignoreCase = true
if not objRegExp.test( thisHome ) then
      thisHome = "HTTP://" & thisHome
end if %>
<td><a href="<%=thisHome%>"><%=thisHome%></a></td>
```
Fig. 6. Code within names.asp that checks the URLs for “HTTP://”

This problem was solved by using a regular expression to examine the record of every URL. The code then adds “HTTP://” only if it is missing. The code for this solution is shown in Fig. 6.

Problems Easily Overlooked

After the development was finished and the code had been successfully deployed, some additional problems emerged. These problems developed from two issues that are easily overlooked in the development of any Web application.

1. Stale Session Variables

The first problem surfaced when users discovered that the links at the top of the page to other class resources did not always work correctly.

It was discovered that the links stopped working only after a user had navigated to the Online Address Book and then left the page open for more than twenty minutes. The conclusion was that the problem was caused when the session expired.

This conclusion was correct, because these links funneled user navigation through a routing script, goNext.asp. This script checked the "CourseID" session variable for the name of the specific section of the current user's class and used it to create the next page request. Whenever the session expired and this session variable was abandoned, the routing information was lost.

The problem was easily solved by storing the course ID in a local variable that is always added to the query string whenever the user navigates to another page.

2. Potential Security Violations

The second problem was identified when students working in the lab would forget to log out from the Address Book after changing their data. This oversight made the student's Address Book data vulnerable if he or she left their work station unattended. A malicious user could surreptitiously sit at the unattended station, navigate back to the Address Book, and alter the data.

This problem was solved by changing the code so that the user is automatically logged out as soon as a change is completed.

Conclusion

The project-oriented approach to teaching Web database development has been very successful at UHCL. The Online Address Book shows that its students are capable of creating a web site that is both effective and innovative.

Reference

[1] The textbook used in this course is: J. Buyens, *Web Database Development Step by Step*. Redmond: Microsoft Press, 2000.

---

[1] Charles R. Moen, M.S., Adjunct instructor at the School of Science and Computer Engineering, University of Houston - Clear Lake, Houston, TX, crmoen@juno.com

[2] Morris M. Liaw, Ph.D., Associate Professor of Computer Science and Computer Information Systems at the School of Science and Computer Engineering, University of Houston - Clear Lake, Houston, TX, liaw@cl.uh.edu