

A General Purpose Messaging System Used to Coordinate Class and Advising in a College of Business

Robin Snyder
Savannah State University
126 Jordan, P. O. Box 20359
Savannah, GA 31404
912-356-2716
snyderr@savstate.edu
<http://www.RobinSnyder.com>

Abstract

In recent years, the use of email messaging as a means of reliable workgroup communication has become more and more problematic. In part, this is due to SPAM, phishing, software security updates, etc. This paper (and talk) will discuss (and present) the design, implementation, and use of a web-based messaging system as part of a larger portal system that is used to coordinate class, advising, and faculty messaging activities in a College of Business. The group/member management is XML-based for flexibility and portability. The actual messages are contained in two SQL tables. The messaging system is contained in two main ASP (or PHP) pages (summary and detail view) and several smaller pages (e.g., for file download, viewing as web page, etc.).

Introduction

This paper will discuss some of the design considerations and implementation details of a web-based messaging system that is in the process of being used for class and advising purposes in a college of business.

History

Email goes back to 1971. The author started using email in graduate school at Penn State in 1984. Since the proliferation of the Internet in the 1990's, many people today use email. Email can be used to communicate and share information using what is called a message passing system. The web supports sharing using what is called a shared memory system. Both message passing and shared memory can be intermixed and any one can emulate the other.

The author began accepting student submissions electronically via a submission system on a CMS/VM IBM mainframe using the author's class management system called Classy starting in 1985 [1], based on a simpler command line system that the author began using in 1984. This worked well at the time. In the 1990's, the author began porting the needed functionality of Classy to the DOS operating system. In the late 1990's, the author moved the system to the Windows platform and began adding the submission functionality that had been missing since the mainframe days. Gradually, more and more of the system has become web-based.

Novell GroupWise did not support email automation very well, but, in 1998, the author began using Microsoft Outlook, which could be automated via the MAPI (Messaging API) interface using VBA (Visual Basic for Applications). Later, using the COM (Component Object Model)

technology, Borland Delphi (i.e., essentially an greatly improved Turbo Pascal for Windows) was used to integrate the class management with email management.

Over the years, as the Internet popularity grew, so did email viruses, Trojan Horses, etc. When Microsoft added security features to an upgrade, it no longer became possible to send outgoing email without either a long delay or a user confirmation for each email. The author switched to using SMTP (Simple Mail Transfer Protocol) to send outgoing email. Then, some Internet Service Providers began introducing delays on outgoing mail. Running VBA macros became more difficult due to security features in Microsoft Office products. And, viruses could always cause problems. In addition, it became hard to make sure that students had and used their email accounts. Some free accounts filtered email, or prohibited certain file attachments. As did some institutional and/or company firewalls. In short, email had become an impediment to inter-organizational communication. The author gradually moved most email functions of the class management software to the web server. This actually worked much better, but some messaging service was still needed.

Workgroup software such as Lotus Notes supports such messaging, but is costly and must be used by the entire organization. The author tried Lotus Notes in 1997 and 1998, but it did not work well without institutional support. Microsoft Exchange and Outlook allow such messaging, but has some of the problems previously mentioned. And, it typically requires institutional support to use Microsoft Exchange.

At the end of 2005, the author decided to add a web-based messaging system to the author's existing web content management system. Such a messaging system has the potential to provide useful, timely, and accurate messaging within the COBA (College of Business). The rest of this paper describes the group and member system, the authentication system, the actual messaging system, and the classroom and advising uses of the messaging system. A brief overview is provided of the rules-based system for prerequisite checks and student advising checks that can be delivered to students and faculty using this system. As such, the messaging system is part of what would be called a portal, but, in this case, localized to the COBA.

Underlying data

The underlying data consists of the following.

- A table of faculty members, including advisor roles.
- A table of students taking COBA classes.
- A table of current classes, teachers, and students.
- A rule/table of advisors for students.

The users are faculty and students. Note that the faculty table actually includes staff support people. The faculty and students are subclasses of larger class of people (i.e., in database terms, an IS-A relationship), but for discussion purposes two separate tables will be used.

XML (Extensible Markup Language) provides a nice way to create and manage much of the user information. XML is quick, easy to work with, and does not require the complexity, overload, licensing, etc., that is needed for a production database. In this case, an off-line database is used to manage the members and groups and the XML is generated when necessary. This is possible

because class rosters and student advisors do not change very often. Automated updates during busy times of the semester are done once or twice daily. Other times updates may be done once or twice a week. Students not taking COBA classes are marked as archived. Students taking a COBA class for the first time are assigned a unique userid and SID (Student Identification Number). For compatibility with the institutional student information system, the SSN (Social Security Number) is maintained locally to match related student information when downloaded but, for security reasons, no SSN is stored on the web server. A SID permits the login name to be changed with propagating changes throughout the distributed database (i.e., local and remote database tables and generated XML).

In the case of the messaging system, two database tables are needed to manage the messages. SQL Server is used as the database. The two tables will be discussed in more detail under the messaging system.

Group and member system

The foundation of any messaging system is the group and member system [3]. A simple approach consists of members with user-created group support. In this case, it is expedient to automatically create some groups from a knowledge of the organization. The class rosters with students and teachers provides one source of groups and members. The advisor assignment rule provides another source of groups and members. Here is an XML fragment for a teacher/advisor.

```
<?xml version="1.0" standalone="yes"?>
<rmsMember
  role="faculty" sid="99974" code="omitted"
  year="faculty" college="COBA" major="MGNT"
  login="smithj" name="Smith, John">
  <group groupId="smithj"
    groupName="Smith, John (you)"
    ownerId="smithj" ownerName="Smith, John"
    type="you" role="you" />
  <group groupId="crn-21018"
    groupName="MGNT 3185 MW 2:30 (teacher)"
    ownerId="smithj" ownerName="Smith, John"
    type="class" role="teacher" />
  <!-- and so on ... -->
  <group groupId="classes-smithj"
    groupName="Students of Smith, John (teacher)"
    ownerId="smithj" ownerName="Smith, John"
    type="students" role="teacher" />
  <group groupId="advise-smithj"
    groupName="Advisees of Smith, John (advisor)"
    ownerId="smithj" ownerName="Smith, John"
    type="advise" role="advisor" />
  <!-- and so on ... -->
</rmsMember>
```

The XML fragment for a student has a similar structure.

Authentication system

In order to provide security and privacy, a secure login system, here using https, is needed so that the system knows who is accessing the server from the client side [8]. The initial approach is as follows.

- Every user is assigned a login name and hard-to-guess randomly generated password.
- The login names and passwords are distributed.
- Once the user logs in, the user can go to "**Preferences**" and set an easy-to-remember password.

An XML file, whose filename is the login name, stores the mapping to the SID. A missing XML file means that the login name is not valid. An XML file whose filename is the SID stores the user information including password, and group membership. An XML file whose filename is the SID (in another directory) stores the user's preferences, including easy-to-remember password, preferred email address, color scheme, etc.

At the same time as the author's system was implemented, based on the existing login system used by the author, the COBA IT support staff created user login accounts based on LDAP (Lightweight Directory Access Protocol). After creating a mapping between the two user account systems, an extension to the author's system allows the user to use their COBA account to access the author's web-based messaging system, and associated portal features.

For simplicity there is one ASP (Active Server Pages) page that manages the login. For LDAP support, there is an additional page in a directory that requires Windows authentication which, if successful, sets the `LOGIN_USER` server variable which is then mapped to the author's userid.

The delay caused by the two separate login systems caused the messaging system to be used for class and advising purposes primarily by the author's 75 students in the Spring of 2006. All the groups and members and user accounts and passwords were there, but there was no easy way to distribute the login information to the students. Unifying the two authentication systems should ease the transition such that the system should be available to all students soon.

After login, the user is presented with many relevant options at the top of the screen-friendly, as opposed to printer-friendly, page. This appears as follows for students using the author's system.

You can get anywhere from here.				home	ideas	coba	ssu	paws	messages
snyderr:	MKTG3179 of Snyder, Robin (teacher) ▼	logout	google:					help	preferences
members	context	images							
calendar	office hours	index	policies	syllabus	rubrics	scores	database	attendance	advising
work due	in class exam	in class work	submit text	submit file	evaluate	participate			
exams	quizzes	topics	asmts	groups	project	suggestions	final	participation	

Once logged in, the user can specify, using a pull-down list, the group to which they wish to be associated with for a given action. Here is a typical group list, with numbers added for reference purposes.

0. pick a group
1. Smith, John (you)
2. MGNT 3185 MW 2:30 (teacher)
3. MGNT 3185 MW 6:00 (teacher)
4. MGNT 4200 MW 1:00 (teacher)
5. MGNT 4200 MW 6:00 (teacher)
6. MGNT 3185 MW of Smith, John (teacher)
7. MGNT 4200 MW of Smith, John (teacher)
8. Students of Smith, John (teacher)
9. Advisees of Smith, John (advisor)
10. MGNT major (faculty)
11. COBA faculty (faculty)

In this example, option 1 is the default group. Options 2, 3, 4, and 5 are individual classes. Options 6 and 7 include all students of this teacher taking a given course. Option 8 include all students taking a class of this teacher this semester. Option 9 is for advisees of this advisor. Option 10 includes all students whose major is the major for which this advisor advises (usually their primary teaching area). Option 11 is for faculty.

The web system displays different information depending on that group specification. Class groups allow access to student rosters for that class, student images (if available), add/drop snapshot histories, prerequisite violations for that class, etc. Advising groups allow access to student advisees, student advising information, etc., for that advising group.

All of these groups and members are automatically generated from class rosters, rules, etc. Future group support will be added as necessary.

Messaging system

The messaging system is fairly simple and consists of two related tables (messages and targets) and code to manage the tables. The complexity in real-world messaging systems arises in making the system efficient for large numbers of users and a large volume of messages. In the case of this web-based messaging system, there are a limited amount of users and the intended use of the system is for safe and efficient internal messaging of information relevant to faculty, staff, and students of COBA.

Here is relevant structure of the `Messages` table.

```
CREATE TABLE [dbo].[Messages] (  
    [MessageId]          [int] IDENTITY (1, 1) NOT NULL ,  
    [ParentId]           [int] DEFAULT 0    NOT NULL ,  
    [SenderId]           [int] DEFAULT 0    NOT NULL ,  
    [MessageType]       [int] DEFAULT 0    NOT NULL ,  
    [Status]             [int] DEFAULT 0    NOT NULL ,  
    [Recalled]           [int] DEFAULT 0    NOT NULL ,  
    [OneToOne]           [int] DEFAULT 0    NOT NULL ,  
    [SenderName]         [varchar] (80)  DEFAULT '' NOT NULL ,  
    [SenderIP]           [varchar] (15)  DEFAULT '' NOT NULL ,  
    [GroupId]            [varchar] (31)  DEFAULT '' NOT NULL ,  
    [GroupName]          [varchar] (40)  DEFAULT '' NOT NULL ,  
    [DateTime]           [datetime]    DEFAULT GETDATE() NOT NULL ,  
    [Subject]            [varchar] (80)  DEFAULT '' NOT NULL ,  
    [FileName]           [varchar] (80)  DEFAULT '' NOT NULL ,  
    [FileType]           [varchar] (40)  DEFAULT '' NOT NULL ,  
    [FileExt]            [varchar] (10)  DEFAULT '' NOT NULL ,
```

```
[FileSize]          [int]          DEFAULT 0 NOT NULL ,  
[Message]          [text]        DEFAULT '' NULL ,  
) ON [PRIMARY]
```

To keep the system simple, the tables are not fully normalized to third normal form. Instead, some copying is done of names to make the system simpler. This is at the expense of later updating, but, in the case of a messaging system, a message, once sent, should not be changed (for audit trail purposes). Most of the field names should be self-explanatory, but here are some remarks.

- Each message has a `MessageId` as a primary key.
- The `ParentId` allows a hierarchy of message threads, such as replies, to be referenced rather than copied when sending message replies (or in discussion threads).
- The `SenderId`, `SenderName`, and `SenderIP` refer to the sender.
- The `MessageType` is the type of the message.
- The `Status` can be active or inactive.
- A `OneToOne` message is a message that appears to the recipient with letting the recipient know who else received the message.
- The `GroupId` and `GroupName` refers to the automatically generated group that was the basis for this message.
- The `Subject`, `DateTime sent`, `Message body`, and `Recalled` fields have the usual/obvious meaning.
- The `FileName`, `FileType`, `FileExt`, and `FileSize` support a single file attachment for each message.

Here are some design considerations.

Rather than let users delete messages, messages are either active or inactive. This lowers support requirements when a user accidentally deletes a message. Each semester, the message system starts afresh from the user's point of view. A future enhancement would allow users to export their messages in a convenient format.

Only single file attachments are supported. Note that the shared memory approach of the system allows one copy of the file to be stored without added complexity. Currently, file attachments are limited to 1MB but that is easily changed. This makes it easy for a faculty member to share a file with the entire class, or all students taking a given course taught by that faculty member. Note: The size limit will be increased for faculty members.

Messages sent `OneToOne` appear as messages from the sender to the recipient. Otherwise, the distribution list can be seen by everyone. Everyone can also see who has opened and who has not opened the message. This is useful in "getting the word out" as everyone can see who has not opened the message. The teacher knows, and students can let their friends who need to know about the message know that they should check their messages.

Here is the relevant structure of the `Targets` table.

```
CREATE TABLE [dbo].[Targets] (  
    [TargetId]          [int] IDENTITY (1, 1) NOT NULL ,  
    [MessageId]        [int] DEFAULT 0    NOT NULL ,  
    [PersonId]         [int] DEFAULT 0    NOT NULL ,  
    [Active]           [int] DEFAULT 0    NOT NULL ,  
    [Opened]           [int] DEFAULT 0    NOT NULL ,  
    [DateTime2]        [datetime] NULL ,  
    ) ON [PRIMARY]
```

- The `TargetId` is the primary key.
- The `MessageId` is the message to which the target refers.
- The `PersonId` is the SID of the person sending the message.
- A message is either `Active` or `inactive`.
- The `Opened` field indicates whether the message was opened. If so it was opened at `DateTime2`.

For simplicity, there are only a few ASP pages that manage the messaging system. In principle, an entire system can be put into just one ASP/PHP page, but it is often useful to divide the system into pages representing the logical structure of the system.

The "**Views**" page provide a list of messages in the various boxes. Currently, the boxes supported are the "**Active**" messages, the "**Inactive**" messages, and the "**Sent**" messages. Future support may include the "**Drafts**" messages (currently done by sending the message to oneself) and additional folders. A future improvement will allow filtering based on groups. A paging and caching mechanism is provided to keep boxes from growing too large. Here is how the "**Views**" page appears.

	Date/Time Sent	From	Group (originator role)	Subject (select to view)	File	
<input type="checkbox"/>	25	2006-01-23 13:55	Snyder, Robin	COBA faculty (faculty)	Test [1/23/2006 1:46:31 PM]	
<input type="checkbox"/>	24	2006-01-23 13:52	Snyder, Robin	COBA faculty (faculty)	Test [1/23/2006 1:46:31 PM]	
<input type="checkbox"/>	22	2006-01-23 13:44	Snyder, Robin	COBA faculty (faculty)	Test [1/23/2006 1:44:47 PM]	
<input type="checkbox"/>	20	2006-01-21 13:53	Snyder, Robin	MKTG 3179 MW 3:00 (teacher)	Test [1/21/2006 1:32:54 PM]	
<input type="checkbox"/>	15	2006-01-18 13:25	Snyder, Robin	MKTG 3179 MW 3:00 (teacher)	Test [1/18/2006 1:25:44 PM]	

In this and other pages, CSS (Cascading Style Sheets) is used with the HTML (Hypertext Markup Language) and JavaScript to provide bubble hints when the mouse is moved over a field, lowering the learning curve while saving valuable screen space.

The "**Message**" page provides a way to view messages and create new messages. Messages can be sent to anyone in the currently selected group. Users can reply either to the "**Sender**" or to "**Everyone**". Sent and viewed messages are read-only. Here is how a new message window might appear.

The screenshot shows a web-based email interface. At the top, a pink header bar contains the text "Message to be sent to (from) Advisees of Snyder, Robin (advisor) [edit:new.]". Below this, the "New message to (from):" field is set to "Advisees of Snyder, Robin (advisor)". To the right of this field are buttons for "Return", "View: Active", "Inactive", and "Sent". A "Send" button is located to the left of the "From" field, which also shows "From: Snyder, Robin to: [0]". Below the "Send" button, there are "Check All" and "Uncheck All" buttons. The main area contains a list of 16 recipients, each with a checked checkbox and a name. The recipients are arranged in four columns. At the bottom right of the list, there is a checked checkbox and the text "Let recipients see other recipients". Below the list is a "Subject:" field containing "Test [4/20/2006 3:10:57 PM]". Below the subject field is an "Attach:" field with a "Browse..." button. At the bottom, there is a text area for "Paste/type your message text here." and a "View Message Text as Web Page" button.

Another page provides a way for the recipient to view the message text as a web page. The RegExp object is used to identify http: and https: links in the message body and convert them to actual links in the message viewed as a web page. A future enhancement will provide a way to identify and list and visit these link from the "Message" page.

Another page provides a way to return a file attachment to the user.

No attempt is made to use newer techniques such as AJAX (Asynchronous JavaScript and XML) to provide interactivity such as found in desktop applications. Instead, the login system, message system and web portal are used primarily as a messaging and delivery system for class and advising purposes.

A future improvement will, on login, take users directly to the message board if they have any messages that have not been viewed. Another improvement will provide discussion topics for the class notes, requirements, policies, etc.

Advising System

The design and implementation of a prerequisite checking system and an advising system have many features in common (some of the author's previous work on this topic is in [2], [4], [5]). The two types of systems use the same rules, but the rules are applied in a slightly different manner. In a prerequisite checking system, the students in the class are checked to insure that they have met the requirements to take that class. In a simple advising system, a student's transcript is checked with the rules to determine the courses that the student needs to graduate and which of those courses are available in the upcoming term or terms, showing the student the available options for those courses (e.g., availability, days and time, etc.). For example, here is the expanded rule fragment for the course BUSA 4126 (the actual syntax is a little more concise).

```
busa4126
  if (acct2101 also acc211) with "C" since "2005C"
  and (acct2102 also acc212) with "C" since "2005C"
  and (busa1105 also bad105) with "C" since "2005C"
  and busa2105 with "C" since "2005C"
  and (econ2105 also eco201) with "C" since "2005C"
  and (econ2106 also eco202) with "C" since "2005C"
  and (math2181 or math1113)
  and (busa2182 was quan2182 also bad331) with "C" since "2005C"
  and (finc3155 also fin320) with "C" since "2005C"
  and (mgnt3165 also man362) with "C" since "2005C" .
```

There are currently over 70 rules that an advisor must know (almost of 600 lines of rules, including one blank line between rules), and more rules seem to be discovered whenever more students use the system. And, the usual errors in the institutional database (including information only in paper form) add to the problem. But, it is better than nothing and acts as an additional check on the (somewhat incomplete and somewhat unreliable) institutional prerequisite checking and graduation audit system.

The result of using the rules with the transcript or class roster (in a manner similar to, but not exactly like, that of the logic language Prolog) is that not only can a "yes" or "no" answer be determined, but the actual part of the rule that is causing a "no" answer can be displayed to the student/advisor/teacher.

The results for each student can be displayed to the student under an available option.

Other options for advising and class management include class rosters for teachers and individual transcript and check sheet information for students (and advisors).

Summary

In summary, this paper has discussed some of the design considerations and implementation details of a web-based messaging system that is in the process of being used for class and advising purposes in a college of business.

References

- [1] Snyder, R. (1992). CLASSY: a system to automate class administration tasks. Proceedings of the 22nd Annual Meeting of the Southeastern Region of the Decision Sciences Institute. Savannah, GA.
- [2] Snyder, R. (1993). A system for assessing student progress. Proceedings of the 29th Annual Meeting of the Southeastern Chapter of The Institute for Management Science. Myrtle Beach, SC.
- [3] Snyder, R. (1994). Automatically managing groups in a network environment. Proceedings of the 22nd Annual Conference of the International Business Schools Computing Association. Baltimore, MD.

[4] Snyder, R. (1995). A system to assist in student advising. Proceedings of the 31st Annual Meeting of the Southeastern Chapter of the Institute for Operations Research and the Management Sciences. Myrtle Beach, SC.

[5] Snyder, R. (1999). Specifying the rules for an advising system with a logic language. Proceedings of the 29th Annual Meeting of the Southeastern Region of the Decision Sciences Institute. Savannah, GA.

[6] Snyder, R. (2000). Classy: An improved system for automating class administration tasks. Proceedings of the 30th Annual Meeting of the Southeastern Region of the Decision Sciences Institute. Wilmington, NC.

[7] Snyder, R., & Shim, C. (2005). Towards some automated ways for assessing the programming ability of students. Proceedings of the 35th Annual Meeting of the Southeastern Region of the Decision Sciences Institute. Raleigh, NC.

[8] Snyder, R. (2006). Using ethical hacking to educate users about secure passwords by cracking insecure passwords using readily available software. Proceedings of the 39th Annual Conference of the Association of Small Computer Users in Education. Myrtle Beach, SC.