

Excel Project

Ward List Emailer

4/13/2011
ISYS 540
Kent Norman

Executive Summary

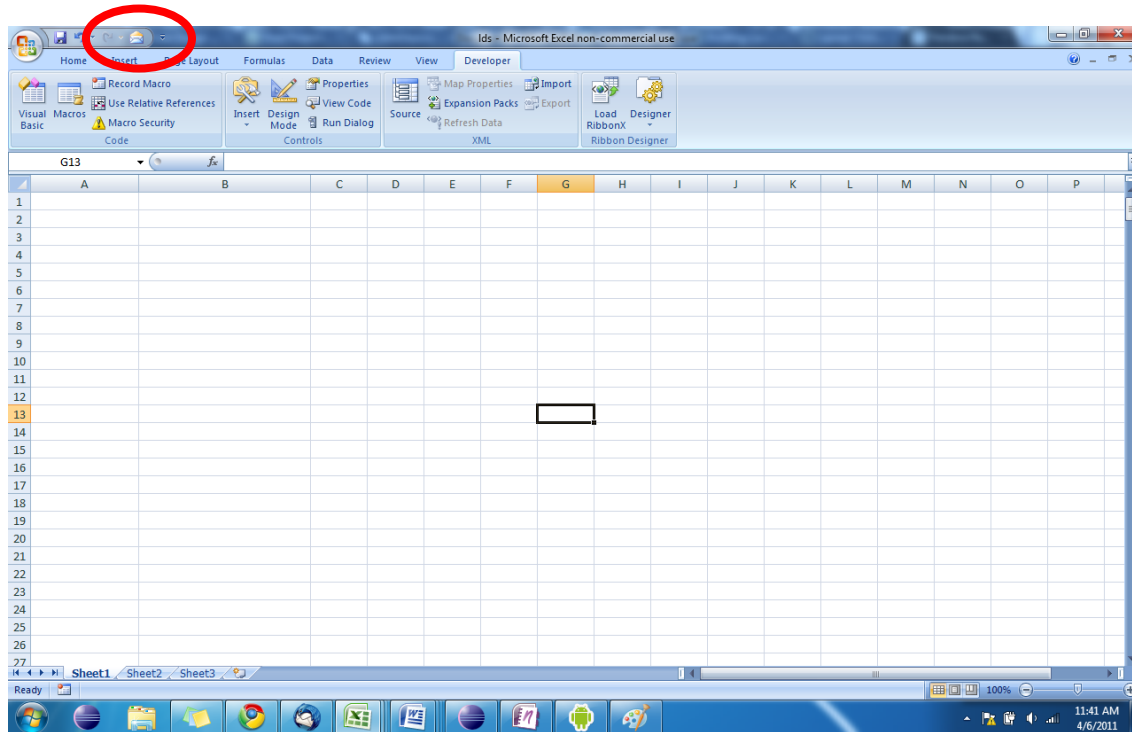
The need for my project came from my activity in the church. I was tired of looking for email addresses of ward members only to forget or lose them once I wrote my email. Excel is a common tooled used around the world and I believe that many members of the church have the same problem that I do. Wards that use my program will no longer need to pass around sheets of paper to gather ward members contact information. As long as the information has been given to the church, my program will create a ward list excel spreadsheet that will have the contact information for all members.

My project resolves the problem of not having member's contact information by retrieving the member records for my ward from the internet to create an excel spread sheet. The information is downloaded to a file location of the user's choice. Once the file is downloaded the user will be able to email other ward members from excel.

Implementation

For my project I implemented a number of techniques and skills that I have learned over the course of the semester. To start the vba program the user will click on the email button that I added in the top left of the screen.

Figure 1 Start



After clicking the button a login screen will pop up. On this form the user will be able to login into lds.org with their username and password. The user will also be able to write the name of the file that they wish to save. This file will be saved in the same directory that the work book is located.

Figure 2 Login Form

A screenshot of a dialog box titled 'Lds.org Login'. It features three text input fields: 'User Name', 'Password', and 'Save file name as:'. A 'Login in' button is positioned at the bottom right of the dialog.

Once the user clicks the login button the login sub begins. This sub uses the agent class that we worked on in class and navigates to lds.org and enters the user name and password that the user entered on the login form to the lds.org website. Once the agent has logged in, the agent navigates to the user's ward site and downloads the ward directory information in a .csv format.

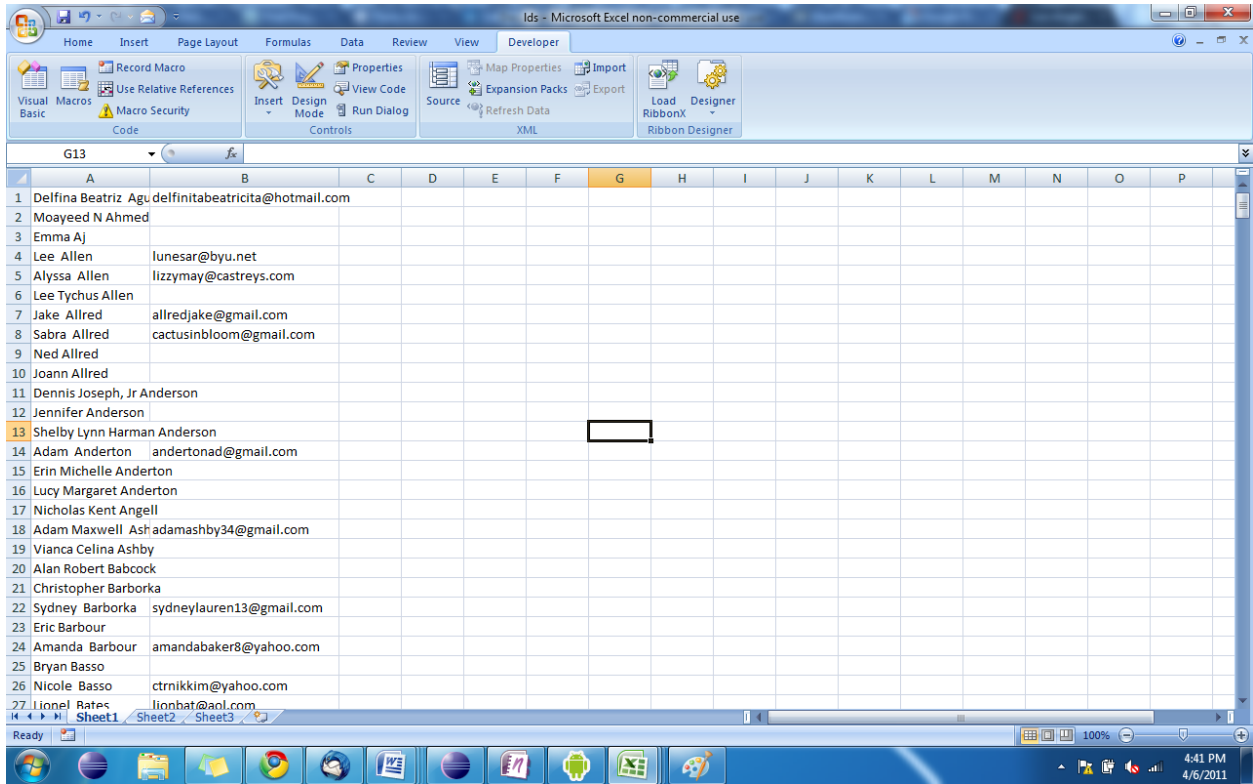
Figure 3 Downloaded Information

The screenshot shows a Microsoft Excel spreadsheet titled 'tellingtest - Microsoft Excel non-commercial use'. The spreadsheet contains data from a CSV file. The columns are labeled as follows: A (familyname), B (phone), C (addr1), D (addr2), E (addr3), F (addr4), G (name1), H (name2), I (name3), J (name4), K, L, M, N, O, P, Q, R. The data rows start from row 1 and go down to row 27. Each row contains a family name in column A, a phone number in column B, an address in columns C-E, and an email address in column G. The email addresses are in the format 'name1 <email>'. The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	familyname	phone	addr1	addr2	addr3	addr4	name1	name2	name3	name4								
2	Aguiar, Delfina Beatriz	801-623-11455 E 600 I	Provo, Utah	84606		test work	Delfina Beatriz	<delfinitabearicita@hotmail.com>										
3	Ahmed, Moayeed N	385-204-8:778 E 560 I	Provo, Utah	84606			Moayeed N											
4	Aj, Emma	801-375-1:1083 E 500	Provo, Utah	84606			Emma											
5	Allen, Lee and Alyssa	970-631-5:427 N 600	Provo, Utah	84606			Lee <lune Alyssa <liz Lee Tychus											
6	Allred, Jake and Sabra	801-493-5:348 N 700	Provo, Utah	84606			Jake <allre Sabra <cactusinbloom@gmail.com>											
7	Allred, Ned and Joann	801-374-1:1034 E 500	Provo, Utah	84606			Ned Joann											
8	Anderson, Dennis Josi	801-472-4:460 N 800	Provo, Utah	84606			Dennis Jo Jennifer											
9	Anderson, Shelby Lyn	760-717-8:1130 E 450 N, # 18	Provo, Utah	84606			Shelby Lynn Harman											
10	Anderton, Adam and	541-310-7:999 E 450 I	Provo, Utah	84606			Adam <an Erin Michele Lucy Margaret											
11	Angell, Nicholas Kent	385 N 800	Provo, Utah	84606			Nicholas Kent											
12	Ashby, Adam Maxwell	480-299-2:419 N 500	Provo, Utah	84606			Adam Ma Vianca Celina											
13	Babcock, Alan Robert	801-891-3:365 N 800	Provo, Utah	84606			Alan Robert											
14	Barborka, Christopher	512-627-6:414 N 600	Provo, Utah	84606			Christoph Sydney <sydneylauren13@gmail.com>											
15	Barbour, Eric and Ama	714-785-7:335 E 400 I	Provo, Utah	84606			Eric Amanda <amandabaker8@yahoo.com>											
16	Basso, Bryan and Nico	562-455-9:445 N 400	Provo, Utah	84606			Bryan Nicole <ctnrnkim@yahoo.com>											
17	Bates, Lionel and Ann	801-377-2:1374 E 400	Provo, Utah	84606			Lionel <lic Anne											
18	Bean, Jeffrey Loring	1:949-412-7:512 E 500 I	Provo, Utah	84606			Jeffrey Lo Lindsay Mary <linrho@hotmail.com>											
19	Beckham, Harrison an	801-635-8:429 N 134I	Provo, Utah	84606			Harrison <Lisa <lisav Ross Nuttall											
20	Bennion, John	801-377-7:554 N 800	Provo, Utah	84606			John											
21	Berg, Eric Logan and C	208-521-1:485 N 110I	Provo, Utah	84606			Eric Logan Camille <Adam Logan											
22	Berneche, Benjamin a	801-373-2:462 N 110I	Provo, Utah	84606			Benjamin Sonia Jay											
23	Bischoff, Kelli	1130 E 450	Provo, Utah	84606			Kelli											
24	Black, Lyndi James	445 N 400	Provo, Utah	84606			Lyndi James											
25	Blacker, Bryan and Au	703-868-4:424 N. 300	Provo, Utah	84606			Bryan <bcl Aubri <guitargirl713@gmail.com>											
26	Blake, Ryan and Nich	435-669-7:445 E 400 I	Provo, Utah	84606			Ryan Nichole Renee											
27	Boody, Robert and Eli	319-830-0:608 E 500 I	Provo, Utah	84606			Robert <b Elizabeth											

After the file is downloaded the sub procedure saves the file as the name listed in the save file as login form in the save directory as the workbook. Once the directory is saved the information is parsed into a useable format. The program is able to add names to the list by parsing the name in the name1/G column and adding the first given name to the parsed family name in the familyname/A column. The downloaded csv file is shown in the figure below. I parse names of individuals into the first column and the emails of the individual into the next column of the original lds workbook. I align the two columns so that the name in column one corresponds to the email in column two. When the name doesn't have an email address I leave the second column blank.

Figure 4 Parsed Information

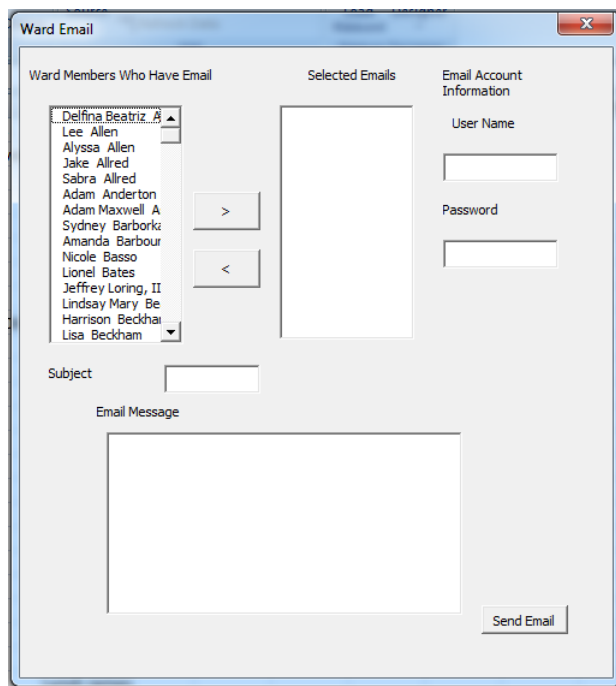


The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Delfina Beatriz Agu	delfinitabatricita@hotmail.com														
2	Moayeed N Ahmed															
3	Emma Aj															
4	Lee Allen	lunesar@byu.net														
5	Alyssa Allen	lizzymay@castreys.com														
6	Lee Tychus Allen															
7	Jake Allred	allredjake@gmail.com														
8	Sabra Allred	cactusinbloom@gmail.com														
9	Ned Allred															
10	Joann Allred															
11	Dennis Joseph, Jr Anderson															
12	Jennifer Anderson															
13	Shelby Lynn Harman Anderson															
14	Adam Anderton	andertonad@gmail.com														
15	Erin Michelle Anderton															
16	Lucy Margaret Anderton															
17	Nicholas Kent Angell															
18	Adam Maxwell	Ashadamashby34@gmail.com														
19	Vianca Celina Ashby															
20	Alan Robert Babcock															
21	Christopher Barborka															
22	Sydney Barborka	sydneylauren13@gmail.com														
23	Eric Barbour															
24	Amanda Barbour	amandabaker8@yahoo.com														
25	Bryan Basso															
26	Nicole Basso	ctrnikkim@aol.com														
27	Lionel Bates	lionbat@aol.com														

After parsing the data the following form is populated.

Figure 5 Email Form



The 'Ward Email' form contains the following sections:

- Ward Members Who Have Email:** A list box containing names: Delfina Beatriz A, Lee Allen, Alyssa Allen, Jake Allred, Sabra Allred, Adam Anderton, Adam Maxwell A, Sydney Barborka, Amanda Barbour, Nicole Basso, Lionel Bates, Jeffrey Loring, II, Lindsay Mary Be, Harrison Bedkha, Lisa Beckham.
- Selected Emails:** An empty list box for chosen recipients.
- Email Account Information:**
 - User Name:** A text input field.
 - Password:** A text input field.
- Subject:** A text input field.
- Email Message:** A large text area for the email body.
- Send Email:** A button at the bottom right.

The list names ward members who have email is populated by the ward members who have an email address. To do this I go through the previously parsed list and check to see if there is an email address to the right of the name. If there is, I add the name to the list on the user form, otherwise the name is skipped.

When a user select a name and then clicks the move arrow, the user name is moved to the selected emails list as shown in the figure below. A name can be removed by selecting the back arrow. When the user clicks sends emails the emails will be sent to each name in this list.

Figure 6 Email Window Filled Out

The screenshot shows a window titled "Ward Email" with a close button (X) in the top right corner. The window is divided into several sections:

- Ward Members Who Have Email:** A list box containing the following names: Lee Allen (highlighted), Alyssa Allen, Jake Allred, Sabra Allred, Adam Anderton, Adam Maxwell A, Sydney Barbork, Amanda Barbour, Nicole Basso, Lionel Bates, Jeffrey Loring, II, Lindsay Mary Be, Harrison Beckha, Lisa Beckham, and Eric Logan Berg. Below the list are two buttons: ">" and "<".
- Selected Emails:** A list box containing the name "Delfina Beatriz Agui".
- Email Account Information:** A section with two labels: "User Name" and "Password". The "User Name" field contains the text "normank". The "Password" field contains a series of asterisks "*****".
- Subject:** A label followed by a text box containing the text "My Grade".
- Email Message:** A label followed by a large text box containing the text "I should get an A on my project".
- Send Email:** A button located at the bottom right of the window.

To be able to send an email the user will need to fill out the other information on the form. The email account information takes the user's Gmail account name and password. This information will be used to send the email through the user's Gmail account. Next the user can enter a subject and the email message itself. Once the user as enter all the desired information and clicks the Send Email button, the emails will be sent. The program then loops through all the select recipients and send them the

message. Upon sending all the emails a message box pops up and tells the user that the messages have been sent.

Learnings and Difficulties

Conceptually the project can be broken down into three separate parts, 1) download the information from lds.org, 2) populate the information in the user forms, and 3) send the emails. Each of these parts of the project provided various learnings and even more difficulties.

1) Downloading information from lds.org

This part proved to be very tricky for me. I spent a number of hours becoming familiar with the agent class that my professor developed. While many of the properties were extended from the internet explorer object, there were a number of features that I learned how to use that were not part of this class or more complex parts of the class. These included the execute script method and the download file method. Both of these methods proved to be very useful for completing my assignment but learning how to successfully manipulate them proved to be difficult. I was grateful for the time that my professor spent working with me to accomplish my goals. I feel that without his help I would still be lost.

2) Populate information in the user forms

Finding a way to parse the information for the emails was a tricky task for me. It was difficult because the email all started at different positions in the cells depending on the length of the name that preceded it. To overcome this problem I used the instr function to identify where the "<" and ">" symbols were located in the text. I then found the difference between the two and used the mid function and started at the number where the instr function found the "<" and continued for the difference that I found in the instr function. This way I could dynamically find all the emails in the worksheet. Once I could parse the emails and the names added that information to the forms was not too bad.

3) Send the emails

This part of the project was the easiest for me. Previously, I had written a program in Java that would send emails. I understood the concepts of smtp servers and ports. Also, the time that our class spent on discussing how to send emails helped reinforce these concepts. The most difficult part was getting the names from the lists and finding the associated email address. I cycled through the rows of data and searched each string to see if it matched the name. If it did I would find the email address on the same row and send the email.