Email List Clean-up

Monte McAllister - December, 2012

Executive Summary

Background

This project is a useful tool to help remove bad email addresses from your many email lists before sending a large batch of emails. It was built for an existing need of a small non-profit publishing organization. They have several lists of email addresses that are used for different purposes. Sometimes these lists overlap and other times multiple lists are combined to send the same email message. Many of these addresses are no longer valid and some people have requested multiple times to stop receiving emails. Unfortunately, the process of keeping these black listed email addresses was all but effective. Each time and email is sent out to a batch, many return emails come back. It has proven to be a headache to the organization and has frustrated many potential customers.

Solution Overview

This solution addresses the problem of cleaning up email addresses. The spreadsheet can be passed around the organization and still access the same blacklist. The functionality is built into the code of the spreadsheet, but no email addresses are saved to the list. The blacklist emails are stored in an access database that will be stored on a server accessible by the members of the organization. Email addresses can be added to the blacklist database using the spreadsheet. But the spreadsheet doesn't need to be saved to a central location.

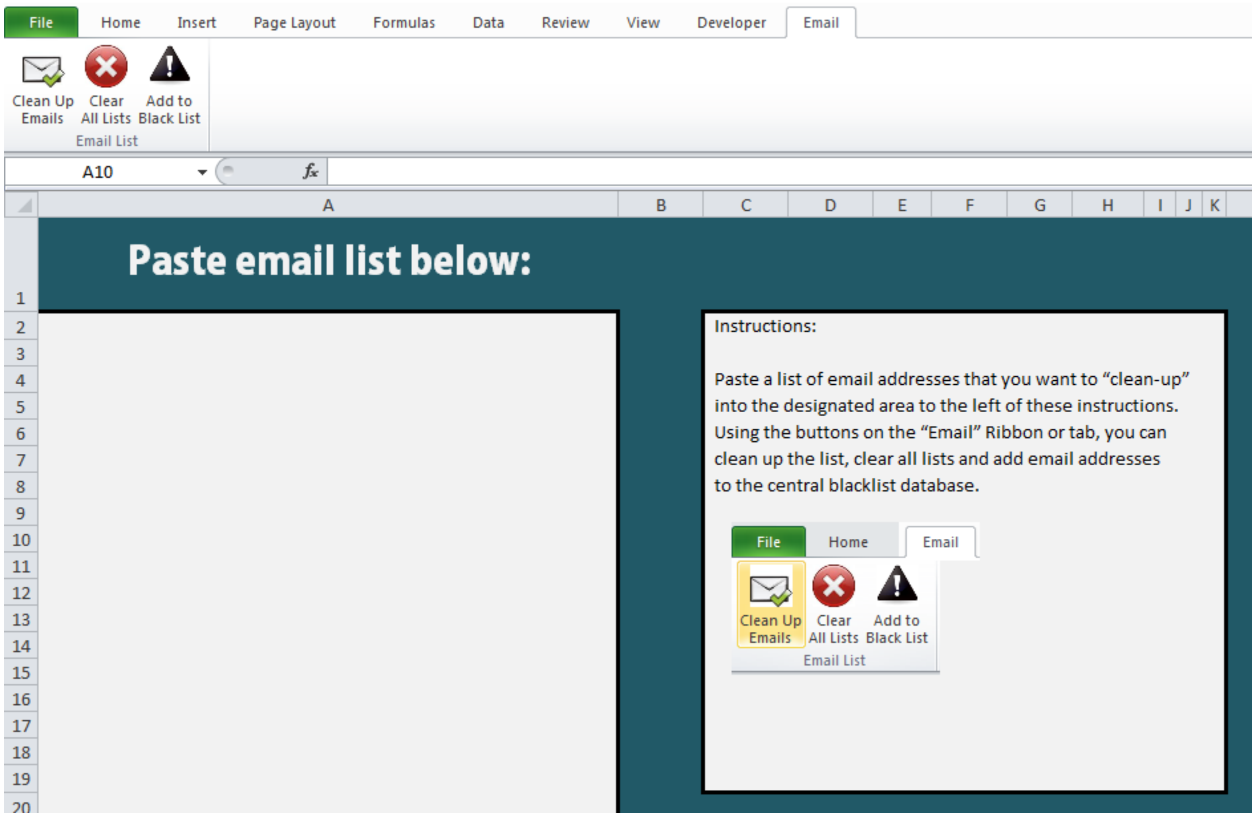
The clean-up functionality takes care of the following problems that have been happening:

- Badly formed email addresses
- Email addresses that have been blacklisted
- Addresses that have requested to unsubscribe from all emails
- Duplicate email addresses from combining lists

The solution will return a cleaned-up email address list that has removed the types of addresses above. It includes a sheet that categorizes the emails that were removed. The cleaned email list that is returned will be sorted alphabetically and all characters will be in lowercase.

Implementation

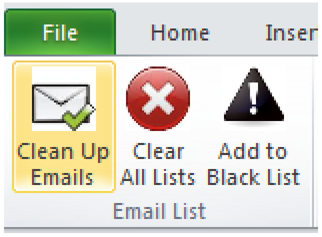
The original idea was to just create a spreadsheet that would ask for two email lists, a blacklist and a dirty email list that needed to be cleaned. That was the first part of the project that was built. But from there, I expanded it to include statistics for the removed email addresses and the central database for storing the blacklist. I made an initial page with instructions to go to the "Email" ribbon to use the macros.



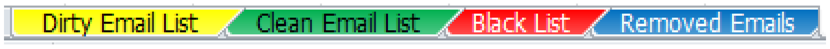
Instructions:

Paste a list of email addresses that you want to "clean-up" into the designated area to the left of these instructions. Using the buttons on the "Email" Ribbon or tab, you can clean up the list, clear all lists and add email addresses to the central blacklist database.

I clearly marked where the email addresses need to be put to be used in this solution. I had to create the buttons on the custom "Email" ribbon and then connect them to different methods in my VBA code.

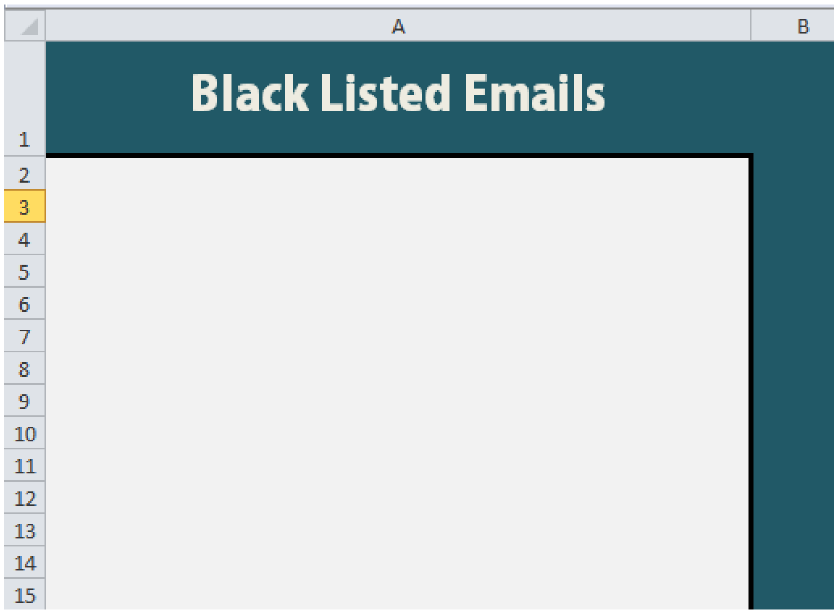


There are three custom buttons that are used in this spreadsheet. I found relevant icons to help the user see what functions the buttons are for. I also color-coded the worksheet tabs.

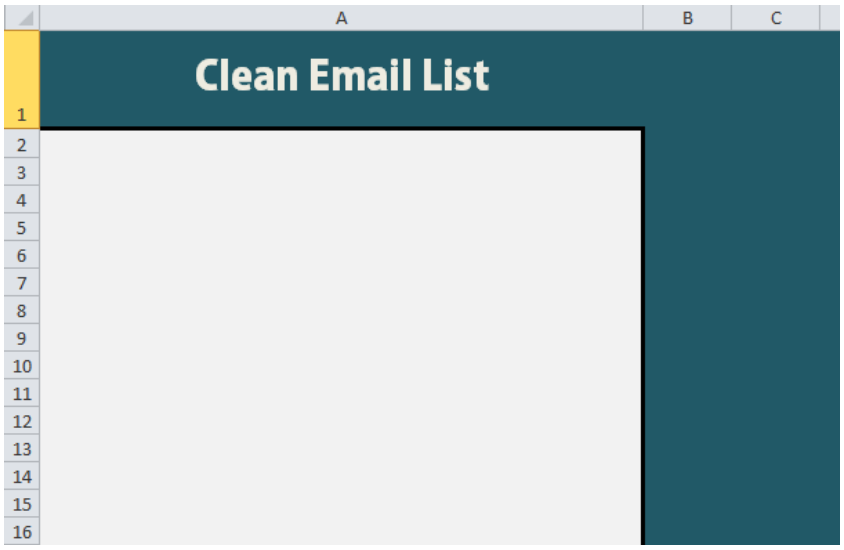


The “Clean Up Emails” button is the main piece of this project. I programmed it to do the following things:

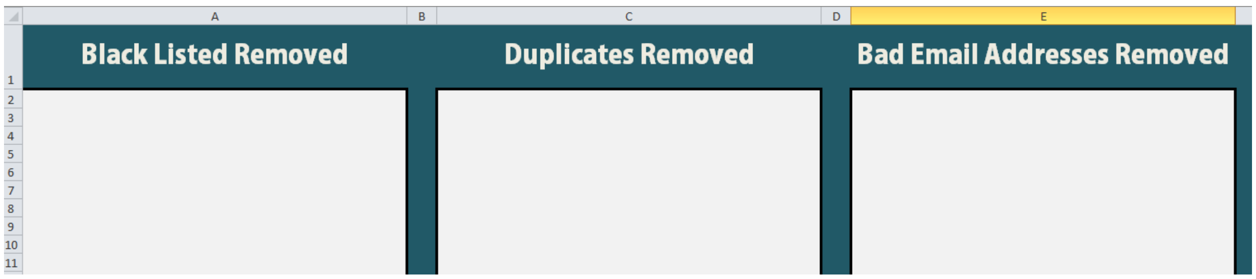
- Take in all email addresses from the dirty list shown above
- Load the black list from the central Access database into the spreadsheet



- Loop through all email addresses and divide them into four arrays in order:
    1. Duplicate email addresses
    2. Invalid email addresses (due to incorrect form)
    3. Blacklisted email addresses
    4. Clean email addresses
- Use the Clean Email address array to populate the clean email list in the spreadsheet



- Use the other email address arrays to populate the removed emails worksheet

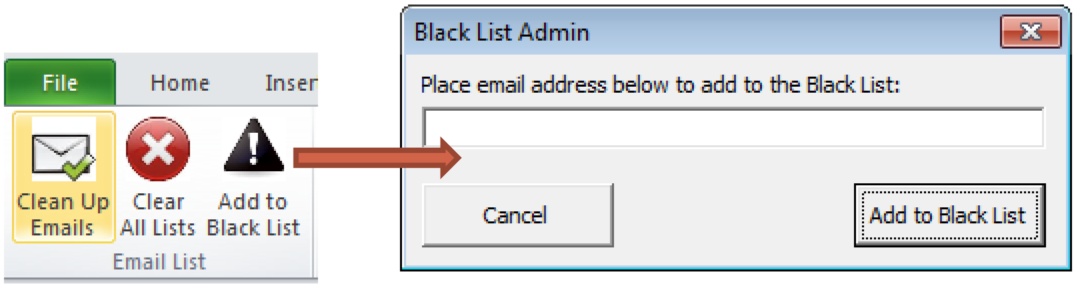


I had to use regular expressions to decide if an email address was formed well enough to be valid. This required adding a reference to the workbook.

Database Manipulation

In class we learned how to insert values to a database. But we didn't write code to read from a database. I had to connect to the database and use a select statement to get the blacklist of email addresses. I then had to step through each one to add it to the spreadsheet. While the database in practice will be in a location accessible to many users, the database included here will be with the spreadsheet file and needs to be in the same directory to have a successful connection.

I added a Black List Admin form to add new email addresses to the database. I specifically only let one address be added at a time to make sure each one is added intentionally.



The final button on the ribbon is the "Clear All Lists" button. This button does what it says and clears all of the email lists in the spreadsheet. It doesn't touch the blacklist database. The main reason for this is to remove addresses from the dirty list so the user can clean up another list. Each time the "Clean-up" method runs, it will clear all of the lists either way.

Learning and Difficulties

I learned many things while building this solution but the biggest and hardest one dealt with the Custom UI Editor that was used to customize the ribbon. I had made some adjustments to my ribbon early on in the project process. I spent many hours building the bulk of my code and saved often. When I was almost finished with the functionality, I decided to make another change to the ribbon. I still had the UI Editor open with my project showing. I made my change and tried to save. It brought up and error because I had the spreadsheet open. So I saved the document and closed it. The UI Editor then saved the ribbon changes without a problem.

Oops! I lost my work. Because I had opened the spreadsheet in the Custom UI Editor before doing all my coding, it still had an old version of the spreadsheet including the few lines of code I had done at that time. There was no way to recover the file and I had to re-do almost the entire project.

Another thing I learned from this is that doing the same thing a second time goes much faster. I was frustrated that I had lost my code, but it was still fresh on my mind. So I re-did a large part of the project in a few hours, the part that had taken many hours to do initially. I now understand the importance of practice when it comes to programming. It's the practice that makes fast coders fast.

Assistance

I used many forums and blogs to help figure out specific code samples for this project. But I was the only one in the class working on it and it is a unique solution as far as I know. I didn't have anybody watching over my work or telling me how to code.

Conclusion

I have enjoyed what I've learned from doing this project. I will probably add to this solution in the future to make it better. I've learned that VBA is a very simple way to do many tasks. Automation is very possible when knowing the power of Excel programming.