Koda Matiaco
4/12/15
Final Project

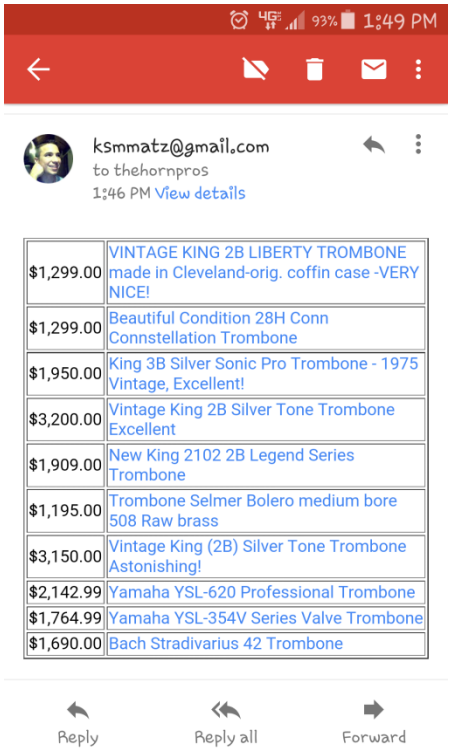
eBay/KSL Scraper

Executive Summary

On a daily basis, I search eBay for musical instrument auctions and bid on them accordingly so I can get a good return when they sell. When I win an auction, I ship the instrument directly to my repairman in Salt Lake City. He repairs them, and sends them to my photographer, who then takes pictures, lists them, and ships them to customers when they sell.

The problem that I have faced since inception of my business is with Buy-it-now options on eBay. Competitors and other users watch eBay closely and buy all of the good fixed-price listings before I have a chance to look at them. With my current job, I have no time to be on the computer looking for deals, so up to this point I have been concerned with auction listings only. There is a lot of money to be made in fixed-price listings, as an example, someone didn't know that their instrument was a collector's horn and sold it for $2500 with the buy-it-now option. In less than a week, it sold for $6250.

The system I built will solve this problem and allow me find good deals on fixed-price listings before anyone else can.



The system scraps an eBay page every 3 minutes and finds if the page has changed since the last time it was checked. If a new listing was posted, the scraper will create a list of the top eBay listings on the page with their respective URL, Title, and Price.

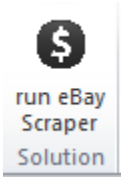
The system will send me an email with the top 10 listings on eBay the eBay page whenever a new listing appears. The email will include a 2-column table with the price, name of the listing, and a URL link connected to the name of the listing. When I get an email from my eBay scraper, I will be able to check the price and title, then click on the link and see if it is a good financial investment.

The notifier works so well with eBay, that I decided to make other programs within this one to scrape KSL for new listings.

User Instructions

- The user first needs to input their information in the “Email” tab of the workbook
    - From Address
    - Email Password
    - To Address
    - eBay URL
    - KSL URL #1
    - KSL URL #2

- This program is meant to scrape an eBay page, the KSL URL’s are optional
- The user will select the button “Start Scraping” or use the custom tab labeled “eBay Scraper” to start the macro.



- If the page has a new listing, the program will send an email to the address specified by the user. The email will look like this:

- The user will see the good deal on top and click on it, look at the pictures, and decide whether or not to buy it.
- This program will check the sites every 3 minutes and continually run without interruption.

Implementation Documentation

- In the code, I used global variables to maintain information for the next time the program runs so that it will be able to compare the listing findings to the previous page information as well as use the information in other sub procedures like “sendMail”.

```
Dim a As New agent
Dim htmlString1 As String
Dim finalHtmlString1 As String
Dim List1(199, 2) As String
```

- I started the sub procedure and declared all of the variables that I would need. Namely, price, name, and link. Upon further work, I found I would need two Boolean variables and a counter for procedural functions.

```
Sub eBayTrombones()

    Dim price As String
    Dim name As String
    Dim link As String
    Dim iterate As Boolean
    Dim first As Boolean
    Dim i As Integer
```

- I set all of my Boolean and counters to a default setting of True and 0. When it runs again, you would start fresh and not have to worry about the code doing funny things based on the previous program that ran (which I had a problem with).

```
htmlString = ""
iterate = True
first = True
i = 0
arrayCounter = 0
```

- The link of the eBay website that I wanted to scrape was set from the worksheet and I used the agent to open the page link to gather the entire HTML results of the page in one place.

```
link = Worksheets("Email").Range("B4")
```

- In case the wifi went out, I wanted my program to continue running without stopping for an error. I added code for error handling which made the program continue despite an error. This will allow me to not worry if my program is not running, and it will give the computer a chance to reconnect to the wireless connection. The Debug.Print statement will allow me to later look at the past errors and troubleshoot when it becomes a problem.

```
On Error Resume Next
a.openpage link
If Err.Number <> 0 Then
    Err.Clear
    Debug.Print "Error opening page #1" & Format(Now(), " mm/dd hh:mm")
End If
```

- The next step is where I got to scrapping the document. I set the position to 1 and moved to various places in the HTML document to gather the link, name, and price of each individual listing. This block of code iterates through all of the listings until the listings on the given page are exhausted.

```
Do While a.moveTo("<div class=""lvpicinner full-width picW"">")
    If Not a.moveTo("<a href=") Then Stop
        link = a.getText(" class=""")
        a.moveTo ("class=""img"" alt='")
        name = a.getText("' />")
        a.moveTo "<span  class=""bold"">"
        price = a.getText("</span>")
```

- If it is the first time running through the while statement, I put the program in an if-elseif-else statement which compares whether the information gathered is new or if no new listings have been added to the list. If no new listings were added, it will trigger the Boolean fields to false and not iterate, nor send an email. I did this because I didn't want the program to do unnecessary processing if the data was already redundant.

```
If first = True Then
    first = False
    If List1(i, 2) = "" Then
        first = False
        iterate = True
    ElseIf List1(i, 2) = link Then
        first = False
        iterate = False
        Exit Do
    Else
        first = False
        iterate = True
    End If

    first = False

End If
```

- If new listings were found, it will continue gathering listing information and set them equal to locations within a list.

```
List1(i, 0) = price
List1(i, 1) = name
List1(i, 2) = link
```

- After the while loop is finished, a For-Next loop prepares HTML to show the results in a table and corresponding links furnished to each of the titles.

```
For i = 0 To 9
    htmlString = htmlString & "<tr><td>" & List1(i, 0) & "</td><td>" & "<a href
Next
```

- After the For-Next loop, I closed off the HTML statement with the proper tags in order to make an html table.

```
finalHtmlString1 = "<table border=""1"" style=""width:100%"">" & htmlString & "</table>"
```

- If the iterate Boolean field is still True, run the sub-procedure “sendMail”. I didn’t want an email every three minutes with the same information, so only if there is a new listing, I would like to receive an email with that information.

```
If iterate = True Then
    sendMail
End If
```

- Within sendMail, local variables are declared. All I needed for this were counter variables to aid in loops if I want to send the email to multiple people.

```
Dim Mail As New Message
Dim Config As Configuration
Set Config = Mail.Configuration
Dim i As Integer
Dim Rows As Integer
```

- Configuration settings such as the SMTPServer, Port, and SSL are set. I also set up the from email and password, corresponding with the excel spreadsheet where you would enter in that information.

```
Config(cdoSendUsingMethod) = cdoSendUsingPort
Config(cdoSMTPServer) = "smtp.gmail.com"
Config(cdoSMTPServerPort) = 465
Config(cdoSMTPAuthenticate) = cdoBasic
Config(cdoSMTPUseSSL) = True
Config(cdoSendUserName) = Worksheets("Email").Range("B1")
Config(cdoSendPassword) = Worksheets("Email").Range("B2")
Config.Fields.Update
```

- I set up the title of the email, along with the HTML for the table to be displayed.

```
Mail.From = Config(cdoSendUserName)
Mail.Subject = "New eBay Listings!" & Format(Now(), " mm/dd hh:mm")
Mail.HTMLBody = finalHtmlString1
htmlString = ""
finalHtmlString1 = ""
```

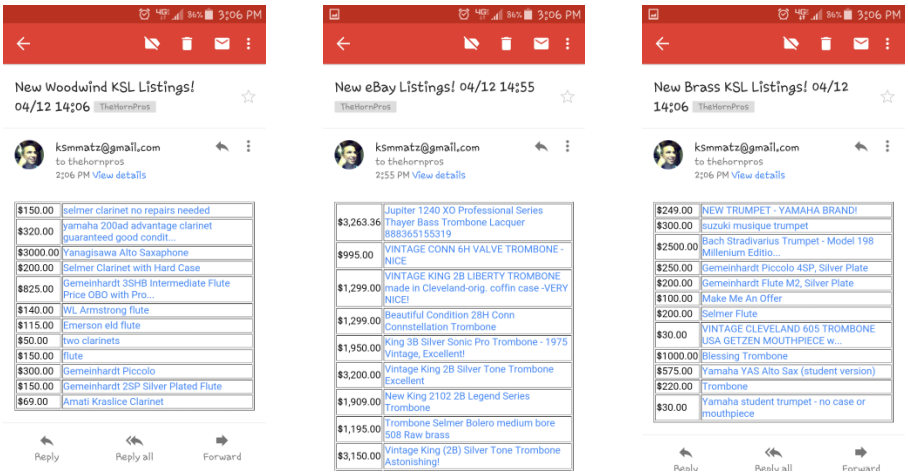
- I set up the “to address” and after that it sends the email of newly listed buy-it-now listings to the specified email address.

```
Mail.To = "thehornpros@gmail.com"
On Error Resume Next
Mail.Send
If Err.Number <> 0 Then
    Err.Clear
End If
```

- Upon completion of the sendMail sub-procedure, the function waits 3 minutes and restarts the process all over again.

```
Application.OnTime Now + TimeValue("00:03:00"), "runProgram"
```

- An email gets sent to my phone and I have the upper advantage of snatching the good deals before anyone else can. The emails come in three varieties; one for eBay, and two others for KSL listings.



Discussion of learning and conceptual difficulties encountered

I learned that if we know the vocabulary, you can find just about anything online. One example with this is when I was looking for a way to have the program run every three minutes. I searched on google “how to run every hour in vba.” I ran into so many different variations on that search but didn’t find the answer. In class, I asked how to do this particular function was called and Dr. Allen said it was the Application.OnTime function. I quickly typed that into google as follows: “vba application.ontime proceed every hour” and the first hit mentioned to add the now() + TimeValue function with the specified amount of time I would like to pass. This goes to show that if I know the terminology, things are so much easier to find online.

One of the toughest things I had to figure out was how to check to see if a new listing was posted. After hours of researching this topic, I found that the best way to figure it out would be to think about it. I scrapped the first listing’s URL, then checked to see if it was the same as the previous URL. I saved the results of the previous RUL by creating a list as a global variable so that I could use it in my “sendMail” sub-procedure. Before this, I was banging my head against the wall thinking about how I could use local variables. Global variables were much easier to use both conceptually and logically. The only problem arises if I want to search multiple pages of eBay while using the same function. I would need to create many more variables to search.

Scrapping a webpage was not as easy as I thought it would be. I searched online and found what seemed to be pretty good tutorials, but they all seemed illogical when I got past the first page of coding. One author stated, “I don’t know why you do this, but you just do.” Dr. Allen

shared with me how he scrapes webpages and it was so much simpler than what was shown online.

I had several errors in the beginning from using SMTP Server Port 22. I didn't know why my program worked at school, and it didn't quite work at home until this question was asked in class. I didn't understand and still don't know why Port 22 was blocked in some instances, but found that Port 465 was a much better option that didn't throw as errors.

I also ran into a problem with errors raising because the wifi would timeout for 10 minutes. Whenever this happened, my program would have an error box show up and stop completely until I clicked "OK". I went through the textbook and found how to enter code for error handling in order to bypass this road block.

Excel can run one macro at a time, but I wanted it to run multiple pages to scrape other pages as well. I simply created a separate macro and ran the individual macros within that one. If I needed anything better than this, I would need to learn a different programming language because VBA wasn't meant for multiple threading.

Assistance

I got most of my code for sending an email here on this youtube video: https://www.youtube.com/watch?v=pFl7W8d7d4M. The only problem was that he used SMTP Server Port 22 and that threw me off a lot, but I eventually found out the problem when the same question was asked in class.

Dr. Allen helped me with scrapping web pages through his vba code that he made. I searched online for assistance and they all proved to be useless and/or confusing. Dr. Allen agreed that that was the case and he happily wrote example code and showed me how to get the information that I needed for my program to start scrapping pages. I then practiced what he showed me on other sites to make sure I had sufficiently learned the material.