Tax Lien Aggregator

MBA 614 - VBA Final Project

Adam J. Hughes
4/13/2012

Executive Summary

Final Project Description

Investors that seek substantial returns by acquiring properties that are distressed usually pay a subscription fee to local firms that provide listings of these properties. These firms typically include title companies, data aggregators, or others that specialize in gathering such information for local real estate professionals.

There are a lot of publicly available websites that provide various types of information regarding distressed properties. These include sites such as www.ksl.com, Zillow.com, realtytrac.com, etc. Most sites with reliable information require a subscription fee. However, most properties that are distressed have property taxes that have not been paid. County websites list those properties that have tax liens against them for back taxes that have not been paid. In Salt Lake County, this website is: http://www.treasurer.slco.org/delqTax/cfml/delinqall.cfm.

The problem with this website is that is provides limited information that in and of itself is not very useful because it doesn't include the address information for the property, only the parcel number. A potential homeowners or investors looking for distressed properties to purchase in Salt Lake County have to access two separate websites in order to determine the address of properties that are subject to a tax sale due to delinquent property taxes. Another website, http://assessor.slco.org/cfml/query/query2.cfm, can be used to find a property's address by entering the parcel number into a search query. This makes it virtually impossible to look for distressed assets in specific geographic areas.

This final project is a web query that aggregates the tax lien information from the first website and address information from the second website. In addition, the second website has an interface with Google Maps that provides coordinate information. This coordinate information includes longitude and latitude that can be used to plot the location of the property in Google Earth Pro.

The query will be limited to those properties that are subject to a tax sale. This will allow a homeowner or investor to visually see the location of all properties in Salt Lake County that will be sold at auction by the taxing authority if the back taxes are not paid in full within five years.

Code Elements

The code that was written for the final project contains three parts: the webQuery sub-procedure to query the county website, a startingRow function that determines where data should start being entered on the query spreadsheet, and the obtainAddress sub-procedure to query the assessor website.

When the county website is queried it lists only 200 properties on each page, and the user has to navigate to the next page to see the next 200 properties. The webQuery sub gathers parcel #, owner name, category code, category description, status, balance due, first delinquent tax year, and tax sale year for each property on each page. It starts on the first page and extracts all the information from that page and stores it on sheet1. It then moves on to the next page and loops to each subsequent page. Each time the query is run, it can download information on 200 properties.

The startingRow function evaluates the row where the last entry was stored from the page last queried and indicates where the first entry of the current page being queried should be entered. The startingRow function allows the webQuery sub-procedure to loop its query on each page of the county website without overwriting information that was already saved on sheet1.

Whereas the county website provides information on 200 properties at a time, the assessor's website provides information on only one property at a time. As such, it takes a significantly longer amount of time to gather address information. Whereas, the county website can be queried in only a few minutes, the assessor's website may take hours. For this reason, the query for the assessor's website is broken out into a different sub-procedure, allowing the user to run each query at different times.

Due to differences in the way the two websites are designed, the obtainAddress sub-procedure needed to make significant adjustments to html text in order to make it useable. That is one reason why the code for this sub-procedure is so long.

Learning Outcomes

I didn't realize when I started this project, that I would have to learn a substantial amount of additional VBA functionalities and HTML properties that were not covered in class. To begin with, the county website presents the results of its query in an html table. I had to learn how to extract information from an html table. I found, http://www.vbaexpress.com/forum/showthread.php?t=31831, which discusses how to do this and provided some code that I used in the final project. I still had to manipulate this code so that it would work on the county website. Figuring this out took the majority of my time on the project.

For the obtainAddress query, I was able to use the code that was written by Professor Allen in the agent.xlsm file for querying websites. I still had to manipulate the code in order to get it to work for the assessor website.

In total, I would estimate that I spent fifty hours on this project. It was very challenging, but also very rewarding.

Results and Next Steps

The queries work beautifully and I was able to download information for 525 properties. Obtaining the addresses took almost two hours, but because it worked so well, I was able to let it run by itself. If I had $300 to purchase Google Earth Pro, the next step would be to import the excel file into Google Earth Pro to see the location of each property within Salt Lake County.

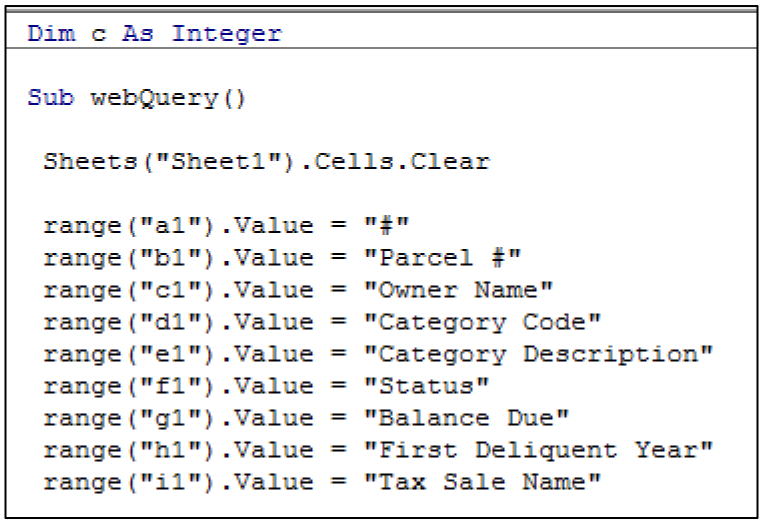
Additional functionality, which I may add in the future, could include removing all parcels with no address from the file and cleaning up the address lines for street types that do not appear in the correct column. Additional query information could also be gathered from the assessor's website including obtaining land record, residence record, tax valuation, adjoining values, neighborhood values, and parcel characteristic information.

Implementation

webQuery Sub-Procedure

Initial Formatting

In order to ensure that the results of the webQuery are not overwriting and mixing with a previous query, the first thing the webQuery sub-procedure does is to clear the contents of sheet1. In order to reformat the blank sheet, the first section of code creates column heading for the webQuery as shown below:

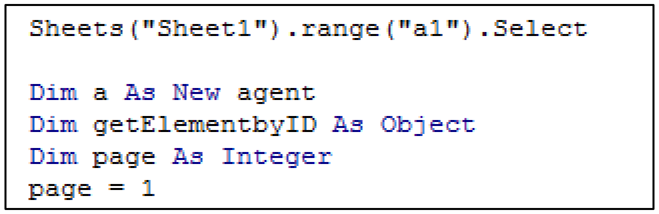


```
Dim c As Integer

Sub webQuery()

 Sheets("Sheet1").Cells.Clear

 range("a1").Value = "#"
 range("b1").Value = "Parcel #"
 range("c1").Value = "Owner Name"
 range("d1").Value = "Category Code"
 range("e1").Value = "Category Description"
 range("f1").Value = "Status"
 range("g1").Value = "Balance Due"
 range("h1").Value = "First Deliquent Year"
 range("i1").Value = "Tax Sale Name"
```

Notice that a module level variable, “c”, is declared. This was necessary in order to store the results of the startingRow function at the module level. This will be described in more detail further on.

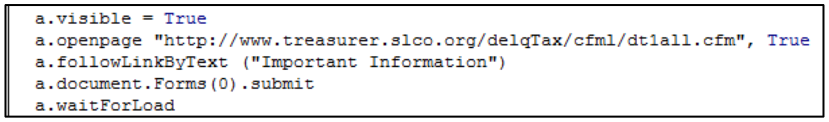
Declaring First Set of Variables

After selecting sheet1 as the active sheet and cell A1 as the starting location for the query, the next step was to declare a few things, beginning with a As New agent. This will allow the code to reference the class module named “agent” that was prepared by Professor Allen. Declaring getElementbyID As Object will allow the code to access a drop down box on the county website and enter a value. A variable “page” was declared and the value was set at one. This will indicate which page of the multi-page (remember that only 200 results are shown on each page) query the code is currently performing operations for.



```
Sheets("Sheet1").range("a1").Select

Dim a As New agent
Dim getElementbyID As Object
Dim page As Integer
page = 1
```

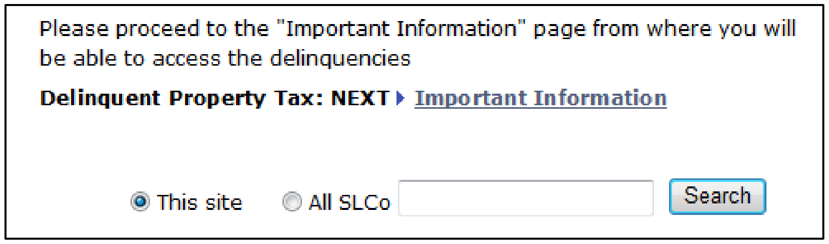
Accessing the County Website



```
  a.visible = True
  a.openpage "http://www.treasurer.slco.org/delqTax/cfml/dtlall.cfm", True
  a.followLinkByText ("Important Information")
  a.document.Forms(0).submit
  a.waitForLoad
```

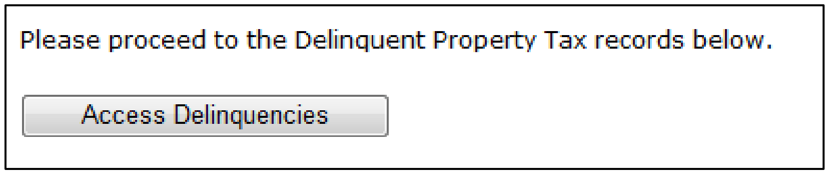
The code above was used to access the county treasurer’s website. This was perhaps the most difficult part of the final project. Because the county treasurer’s website has some sort of sessions functionality built into it, I couldn’t access the query site directly from VBA. I first had to navigate to two other website before I could get to the query site. The initial website’s address is in the second line of code. I then navigated to a hyperlink attached to the

"Important Information" text which led me to another site. See the screenshot below for what this hyperlink looks like:


Please proceed to the "Important Information" page from where you will be able to access the delinquencies

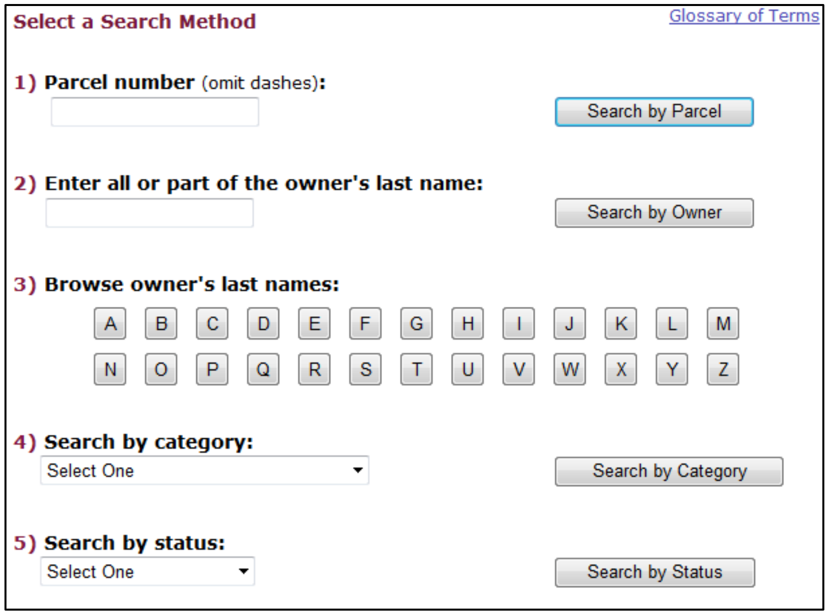
Delinquent Property Tax: NEXT ▸ Important Information

The next website had the exact same address as the first one except it ended with "dt2all.cfm" instead of "dt1all.cfm". For some reason, which I couldn't figure out, I had to access the "dt1all.cfm" website before I could navigate to the "dt2all.cfm" website. At the bottom of this website was a button with the text "Access Delinquencies" as shown below:


Please proceed to the Delinquent Property Tax records below.

I couldn't figure out how to navigate to this button and click on it, given the code in the agent.xlsm file that Professor Allen had prepared, so I had to do some searching on the internet and found that I could use the line "a.document.Forms(0).submit" in order to navigate to the query website.

Finally, I figured out how to get to the query website which is shown below:


Select a Search Method

Glossary of Terms

1) Parcel number (omit dashes):

2) Enter all or part of the owner's last name:

3) Browse owner's last names:

4) Search by category:

5) Search by status:

This interface is pretty useless if you're trying to download the entire database of information that the county offers. There are really three options to get the information. The first would involve browsing by every letter of an owner's last name. The second would involve doing a search by every category. The third would involve searching for property by status. As I am interested in only those parcels that are subject to a tax sale, the last option is the one that makes the most sense. The code below is what I used to select "Tax Sale Certified" from the "Search by Status:" box and submit the request:

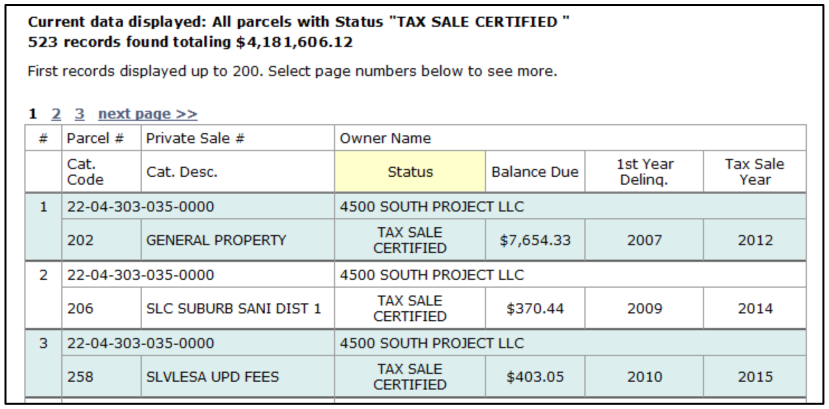
```
a.document.getElementbyID("formStat").Value = "TAX SALE CERTIFIED  "
a.document.getElementbyID("formStat").Focus
Application.SendKeys "{TAB}", True
Application.SendKeys "~", True
a.waitForLoad
a.waitForLoad
```

Notice that I had to use the .SendKeys function in order to tab over to the submission box and submit it using the enter key, which is represented by the symbol ~. I couldn't' figure out how to navigate to this submission box and activate it by using the html page source code. This code has not had any difficulty in navigating to the query page and selecting the appropriate status to search for.

Declaring a Second Set of Variables

```
Start:

 Dim HTMLdoc As HTMLDocument
 Dim TDelements As IHTMLElementCollection
 Dim TDelement As HTMLTableCell
 Dim Spanelements As IHTMLElementCollection
 Dim Spanelement As HTMLTableCell
 Dim r As Long
 Dim number As Integer
 Dim parcel As String
```

The code above begins with "Start:", which is referenced from a piece of code at the end of the sub-procedure. This is the location in the code where the sub-procedure starts-over for a subsequent page of results (only 200 results are shown on each page). Following this are the declaration of some other items. I couldn't even begin to explain what the first five declarations are doing. I gathered this code from the following website: http://www.vbaexpress.com/forum/showthread.php?t=31831.
What I understand from this code is that the query results appear in an html table as shown below:



Current data displayed: All parcels with Status "TAX SALE CERTIFIED "
523 records found totaling $4,181,606.12

First records displayed up to 200. Select page numbers below to see more.

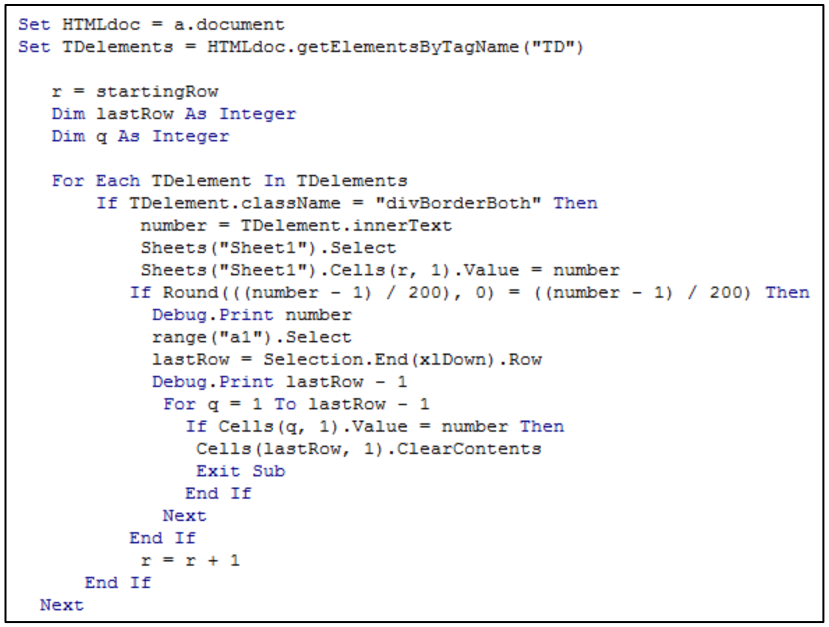
1 2 3 next page >>

#	Parcel #	Private Sale #	Owner Name			
	Cat. Code	Cat. Desc.	Status	Balance Due	1st Year Delinq.	Tax Sale Year
1	22-04-303-035-0000		4500 SOUTH PROJECT LLC			
	202	GENERAL PROPERTY	TAX SALE CERTIFIED	$7,654.33	2007	2012
2	22-04-303-035-0000		4500 SOUTH PROJECT LLC			
	206	SLC SUBURB SANI DIST 1	TAX SALE CERTIFIED	$370.44	2009	2014
3	22-04-303-035-0000		4500 SOUTH PROJECT LLC			
	258	SLVLESA UPD FEES	TAX SALE CERTIFIED	$403.05	2010	2015

There are tag names within the html code called "td" tags and "span" tags. The declarations above are to set a collection of these tags which will be searched through later on in the code.

Extracting Item Numbers

The first data item to download from the query is the item number, which will range from 1 to 200, depending on the number of properties that are returned from the search query. The next section of code downloads these items numbers:



```
Set HTMLdoc = a.document
Set TDelements = HTMLdoc.getElementsByTagName("TD")

   r = startingRow
   Dim lastRow As Integer
   Dim q As Integer

   For Each TDelement In TDelements
       If TDelement.className = "divBorderBoth" Then
           number = TDelement.innerText
           Sheets("Sheet1").Select
           Sheets("Sheet1").Cells(r, 1).Value = number
          If Round(((number - 1) / 200), 0) = ((number - 1) / 200) Then
            Debug.Print number
            range("a1").Select
            lastRow = Selection.End(xlDown).Row
            Debug.Print lastRow - 1
             For q = 1 To lastRow - 1
               If Cells(q, 1).Value = number Then
                Cells(lastRow, 1).ClearContents
                Exit Sub
               End If
             Next
          End If
           r = r + 1
      End If
  Next
```

I'm not entirely sure what the first two lines of code are doing, but the next line down calls the startingRow function, which determines what row the data should be entered in on the sheet1 worksheet. The variable r is set to equal this starting ro. The next section of code is a loop that goes through each tag labeled "td" in the html source and selects those "td" tags with a class name of "divBorderBoth" and returns the text that is within this tag, which is the item number. These get entered into column A on sheet1. The debug.print lines are just to see if the code is operating correctly.

Ending the webQuery Sub-Procedure

The next section of code is very important and was necessary in order to stop the sub-procedure from continuing to re-enter the query results from the last page of the query. I'm sure there is another way to do this, but I couldn't figure it out. What I did here was to see if the first item # on the current query page had already been entered in sheet1, and if it had already been entered, then it stops the sub-procedure from continuing.

As the first item # on the query pages will be a 1, 201, 401, 601, etc., the code checks first to see if the current item it is downloading is the first item # on a query page. The "If Round" section of the code performs this function. If the current item is the first item # on a query page the code within the if function continues to operate. It's next step is to determine whether this item # has already been recorded on sheet1. It does this by determining the last row of data that has been entered and saving it as the variable "lastRow". It then performs a loop to determine whether the current item # has already been entered on sheet1.

However, because the current item # is first save to the worksheet before this check is performed, the code was altered to look through the range of items beginning with row 1 all the way through the second to last row. If the current item # is found within this range, it means that the sub-procedure has already performed a query on this page and downloaded information for every item #. If the current item # is found, the code exits the sub-procedure and deletes the contents of the last row.

The last section of code i"r=r+1", moves the code onto the next empty row for the next item # and loops back to the top of the For Each loop.

Downloading the Parcel Number

```
  'the following code will download the Parcel number

Set HTMLdoc = a.document
Set Spanelements = HTMLdoc.getElementsByClassName("textAlignLeft")

r = startingRow
   For Each Spanelement In Spanelements
    If Spanelement.className = "textAlignLeft" Then
      parcel = Spanelement.innerText
            If InStr(1, parcel, "-", vbTextCompare) > 0 And _
            IsNumeric(Mid(parcel, 1, 1)) = True Then
             Sheets("Sheet1").Cells(r, 2).Value = parcel
            End If
      End If
     If Not Sheets("Sheet1").Cells(r, 2).Value = "" Then
      r = r + 1
     End If
    Next
```

The next snippet of code downloads the parcel number. As previously mentioned, the startingRow function determines what row the data should begin to be entered on in the sheet1 worksheet. This is found in all subsequent sections of code that download data from the query pages, so it won't be mentioned again to avoid needless repetition.

The code for downloading the item # is similar to this code except that a "span" tag was used in the html code instead of a "td" tag. The class name was different as well, with "textAlignLeft". As a parcel number is entered the format, "XX-XX-XXX-XXX-XXXX", The span tags were then searched to return only the inner text that had the symbol "-" in it.

However, this didn't eliminate all the inner text that wasn't a parcel number. The next line of code also checks to make sure that the first character in the string is a number. If these two checks are affirmative, then the string is saved on sheet1 in column 2. As not all the strings within the "span" tags are parcels, this left spaces in column 2 for these items.

The next section of code checks to see if a value was stored in sheet1 for the current "span" tag it is evaluating. If no value was entered and the cell is blank, then the code does not execute "r=r+1" to move down to the next row. Instead it stays on the current row until it finds a parcel number to enter into the cell. This ensures that there are no empty spaces in column 2 and that the item #'s match up with the corresponding parcel numbers.

Download the Owner Name

```
       'the following code will download the Owner Name

r = startingRow
Dim owner As String
    For Each TDelement In TDelements
        If TDelement.colSpan = "4" Then
            owner = TDelement.innerText
            Sheets("Sheet1").Select
            If Not owner = "Owner Name" Then 'new
            Sheets("Sheet1").Cells(r, 3).Value = owner
             If Not Sheets("Sheet1").Cells(r, 3).Value = "" Then
              r = r + 1
             End If 'new
             End If
        End If
    Next
```

This was the easiest data to extract from the html code because of the way it was encoded. The code is very similar to the other item # and parcel number code. However, instead of using class name to determine the desired text, colSpan was used. Each Owner Name was encoded to colSpan 4, making this rather easy to download. The only difference, is that this code would return the column heading, "Owner Name", which has already been entered on sheet!.

So the code was written to exclude this value from being entered on sheet1. The last section of code does the same thing as the last section of code for the parcel number. That is, making sure that no blank cells are created.

Downloading the Category Code

```
  'the following code will download the category code

r = startingRow
Dim catcode As String
   For Each Spanelement In Spanelements
   If Spanelement.className = "textAlignLeft" Then
    catcode = Spanelement.innerText
         If Not InStr(1, catcode, "-", vbTextCompare) > 0 And _
         IsNumeric(Mid(catcode, 1, 3)) = True Then
             Sheets("Sheet1").Cells(r, 4).Value = catcode
            End If
      End If
   If Not Sheets("Sheet1").Cells(r, 4).Value = "" Then
    r = r + 1
   End If
  Next
```

This code snippet is pretty straightforward. It's similar to the other code items listed above, with little variation. There's nothing really new with this code.

Downloading the Category Description

```
 'the following code will download the category description

 r = startingRow
 Dim catdesc As String
  For Each Spanelement In Spanelements
  If Spanelement.className = "textAlignLeft" Then
   catdesc = Spanelement.innerText
       If IsNumeric(Mid(catdesc, 1, 1)) = False Then
            Sheets("Sheet1").Cells(r, 5).Value = catdesc
           ' r = r + 1
       End If
     End If
```

Again, this is similar to the other code that's been written thus far for the other data items. The only difference is that the code would return the column headings for all the items, so the following code was written to exclude these from being entered on sheet1:

```
 If Not Sheets("Sheet1").Cells(r, 5).Value = "" Then
 If Not Sheets("Sheet1").Cells(r, 5).Value = "Parcel #" Then
 If Not Sheets("Sheet1").Cells(r, 5).Value = "Private Sale #" Then
 If Not Sheets("Sheet1").Cells(r, 5).Value = "Owner Name" Then
 If Not Sheets("Sheet1").Cells(r, 5).Value = "Cat. Code" Then
 If Not Sheets("Sheet1").Cells(r, 5).Value = "Cat. Desc." Then
  r = r + 1
 End If
 End If
 End If
 End If
 End If
 End If
Next
```

Downloading the Status

```
   'the following code will download the status

   r = startingRow
   Dim status As String
   Set HTMLdoc = a.document
Set TDelements = HTMLdoc.getElementsByTagName("TD")

   For Each TDelement In TDelements
       If TDelement.className = "divBorderBottom" And _
       TDelement.Align = "center" Then
           status = TDelement.innerText
           If IsNumeric(Mid(status, 2, 1)) = False Then
           Sheets("Sheet1").Select
           Sheets("Sheet1").Cells(r, 6).Value = status
           r = r + 1
           End If
       End If
   Next
```

This code is again, pretty similar to what's been written so far, so nothing really new here. The only difference is that it excludes all inner text items in "td" tags that are numeric. This is because the code was including some non-desirable text that wasn't related to status. This eliminates those items.

Downloading the Balance Due

The following code downloads the outstanding taxes that are due. This was pretty simple to write, using the code that I had written so far, and doing some minor changes. By searching for inner text items that began with a dollar sign ($), I was able to isolate the desired text. See code below.

```
   'the following code will download the balance due

       Dim balancedue As String

   Set HTMLdoc = a.document
Set TDelements = HTMLdoc.getElementsByTagName("TD")

  r = startingRow
   For Each TDelement In TDelements
       If TDelement.className = "divBorderBottom" And _
       TDelement.Align = "center" Then
           balancedue = TDelement.innerText
           If Left(balancedue, 1) = Chr(36) Then
           Sheets("Sheet1").Select
           Sheets("Sheet1").Cells(r, 7).Value = balancedue
           r = r + 1
           End If
       End If
   Next
```

Downloading the First Delinquent Year

```
'the following code will download the first delinquent year

       Dim delinq As String

    Set HTMLdoc = a.document
 Set TDelements = HTMLdoc.getElementsByTagName("TD")

    r = startingRow

    For Each TDelement In TDelements
        If TDelement.className = "divBorderBottom" And _
        TDelement.Align = "center" Then
            delinq = TDelement.innerText
            If Not Left(delinq, 1) = Chr(36) Then
            If IsNumeric(Mid(delinq, 2, 1)) = True Then
            Sheets("Sheet1").Select
            Sheets("Sheet1").Cells(r, 8).Value = delinq
            r = r + 1
            End If
            End If
        End If
    Next
```

Unfortunately, the way the html code was written, the first delinquent year and tax year sale (two separate data fields) could not be downloaded separately from the website. As these items were found within "td" tags, class names of "divBorderBottom", and align of "center", they had to be downloaded concurrently and stored on sheet 1 in column 8.

In fact, the balance due item also had the same html encoding but because they could be distinguished by the $ sign (which is represented as Chr(36) in ASCII as noted above in the coding) they could be isolated or in this case, eliminated from the query results for the first delinquent tax year and tax year sale.

Because the resulting download to sheet1 in column 8 had the first delinquent tax year and the tax year sale in the subsequent row, the following code was written to move the value in every other cell over to column 9, adjacent to the cell in column 8 that the first delinquent tax year value was recorded in:

```
    'the following code moves the tax year sale
    r = ((page - 1) * 200) + 2
  Cells(r, c - 1).Select
Do Until ActiveCell.Offset(3, 0) = ""
    ActiveCell.Offset(1, 0).range("A1").Select
    Application.CutCopyMode = False
    Selection.Cut
    ActiveCell.Offset(-1, 1).range("A1").Select
    ActiveSheet.Paste
    ActiveCell.Offset(2, -1).Select
Loop
```

Notice that this is the section of the code where the variable "page" is used to determine the row (variable r) that the code should start on (see first line of code).

This works for moving every cell, but because the code terminates when the cell three rows down is blank, it terminates before moving the very last tax year sale entry. The following code was written to account for this discrepancy:

```
'the following code fixes the last entry
'of the delinquent tax year
 ActiveCell.Offset(1, 0).range("A1").Select
    Application.CutCopyMode = False
    Selection.Cut
    ActiveCell.Offset(-1, 1).range("A1").Select
    ActiveSheet.Paste
```

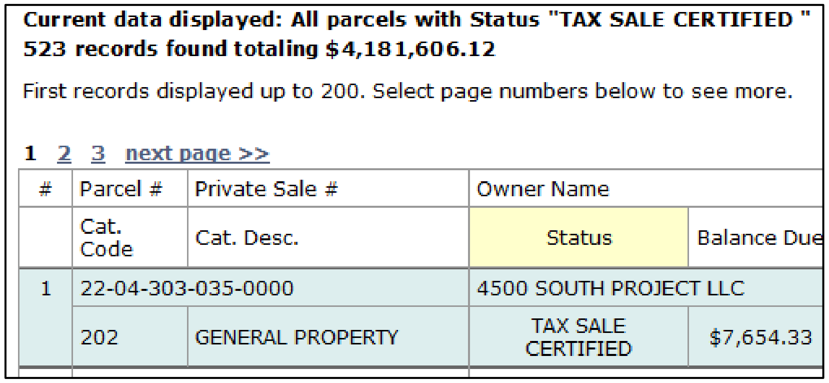
There is still one remaining problem, and that is, there are now blanks in column 8, where the tax year sale values used to be. Because these values had been moved over to the adjacent cell in column 9, the same rows in column 8 and column 9 are blank. The following code was written to copy and paste the values in column 8 and 9 so that there were no blank cells and that the values in these columns corresponded to the data in columns 1 through 7:

```
'the following code removes the empty spaces in
'the delinquent tax year and tax year sale columns

Dim z As Integer
z = 0
For z = 0 To 1
Cells(r, c - 1).Select
ActiveCell.Offset(0, z).Select
Do Until Selection.End(xlDown).Value = ""
    Selection.End(xlDown).Select
    Selection.Cut
    Selection.End(xlUp).Select
    ActiveCell.Offset(1, 0).range("A1").Select
    ActiveSheet.Paste
Loop
Next
```

Notice the use of variables, r, c, and z. The variable z was used to create a loop that would run the code for column 8 and then for column 9. Variables r and c were used because these values had already been determined previously in the code to determine the starting row on which each web query page was to be downloaded. By using these variables, and there corresponding values, the code will execute on the items that had just been downloaded to the sheet1 worksheet and needed to be separated out into column 9 and have the empty spaces removed.

Navigate to Next Query Page

After all the data on the current query page is downloaded, the following code instructs the interpreter to move on to the next query page. Notice the "next page" hyperlink on the web page appears as follows:



Current data displayed: All parcels with Status "TAX SALE CERTIFIED "
523 records found totaling $4,181,606.12

First records displayed up to 200. Select page numbers below to see more.

1 2 3 next page >>

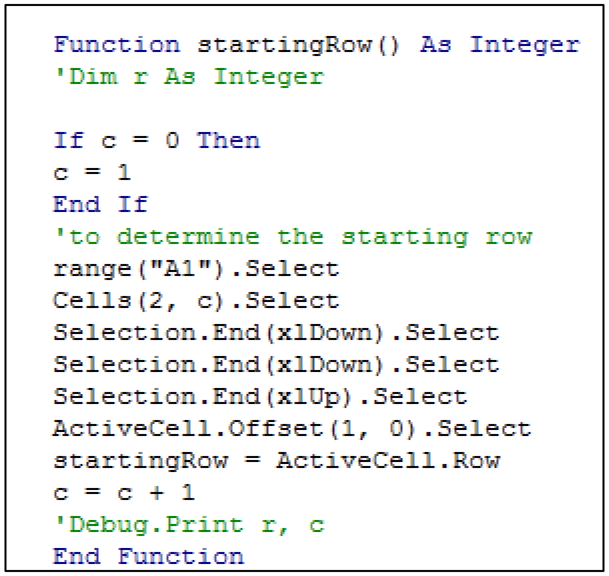
#	Parcel #	Private Sale #	Owner Name	
	Cat. Code	Cat. Desc.	Status	Balance Due
1	22-04-303-035-0000		4500 SOUTH PROJECT LLC	
	202	GENERAL PROPERTY	TAX SALE CERTIFIED	$7,654.33

The code instructs the interpreter to continue to the next page by selected the "next page" hyperlink. It then adds one value to the page and resets c (a variable used to determine what column the interpreter should begin saving data to on sheet1) to 1.

```
'navigate to next page:
Do
a.followLinkByText ("next page")
a.waitForLoad
c = 1
page = page + 1
GoTo Start:
Loop



     End Sub
```

I'm not so sure that the Do Loop is necessary here, as the GoTo instruction moves to the Start: location at the top of the code. This Do Loop is really just a remnant of a loop that wasn't effective at terminating the sub-procedure. With the GoTo command, the interpreter is sent to the Start: location. Code within the item # download section (see above) terminates the sub-procedure.

startingRow Function



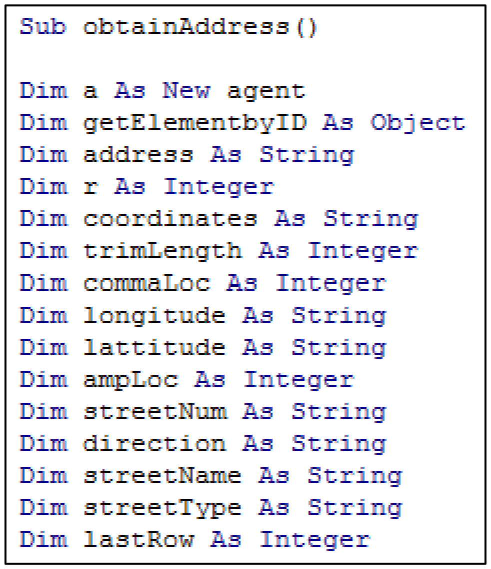
```
Function startingRow() As Integer
'Dim r As Integer

If c = 0 Then
c = 1
End If
'to determine the starting row
range("A1").Select
Cells(2, c).Select
Selection.End(xlDown).Select
Selection.End(xlDown).Select
Selection.End(xlUp).Select
ActiveCell.Offset(1, 0).Select
startingRow = ActiveCell.Row
c = c + 1
'Debug.Print r, c
End Function
```

The code above is for the startingRow function that evaluates the sheet1 workbook to determine the very last row and column that data had been entered on. It then adds one value to each and stores the row as "starting row" and the column in variable "c", a module level variable. As previously discussed, the "starting row" function is called each time a new section of the code is run for downloading a new data item from the county treasurer's website.

obtainAddress Sub-Procedure

As previously mentioned, this sub-procedure accesses the Salt Lake County Assessor's website and performs a search for property addresses and coordinates using parcel numbers. It then downloads the results into the sheet1 workbook.

Declaring Variables



```
Sub obtainAddress()

Dim a As New agent
Dim getElementbyID As Object
Dim address As String
Dim r As Integer
Dim coordinates As String
Dim trimLength As Integer
Dim commaLoc As Integer
Dim longitude As String
Dim lattitude As String
Dim ampLoc As Integer
Dim streetNum As String
Dim direction As String
Dim streetName As String
Dim streetType As String
Dim lastRow As Integer
```

The first section of code declares a lot of variables used throughout the sub-procedure. The first two variables are similar in name and function to those declared in the webQuery sub-procedure.

Formatting and Begin For Loop

```
  Sheets("sheet1").Select
  Sheets("sheet1").range("j1").Value = "Street Number"
  Sheets("sheet1").range("k1").Value = "Direction"
  Sheets("sheet1").range("l1").Value = "Street Name"
  Sheets("sheet1").range("m1").Value = "Street Type"
  Sheets("sheet1").range("n1").Value = "Lattitude"
  Sheets("sheet1").range("o1").Value = "Longitude"

 Sheets("sheet1").range("a1").Select
 lastRow = Selection.End(xlDown).Row
 r = 2

 For r = 2 To lastRow
```

This section of code creates column headings for the address and coordinates. In addition, it determines the last row of the existing data (downloaded by the webQuery sub-procedure in columns 1 to 9) and creates a For Loop beginning at row 2 (the row directly below the column headings) and ending with the last row of data. In this way, the interpreter will search for address and coordinate data for each row of data.

Duplicate Parcel Numbers

```
'to check to see if there are duplicate addresses:
If Sheets("sheet1").Cells(r, 2).Value = Sheets("sheet1").Cells(r - 1, 2).Value Then
 Sheets("sheet1").Cells(r, 10).Value = Sheets("sheet1").Cells(r - 1, 10).Value
 Sheets("sheet1").Cells(r, 11).Value = Sheets("sheet1").Cells(r - 1, 11).Value
 Sheets("sheet1").Cells(r, 12).Value = Sheets("sheet1").Cells(r - 1, 12).Value
 Sheets("sheet1").Cells(r, 13).Value = Sheets("sheet1").Cells(r - 1, 13).Value
 Sheets("sheet1").Cells(r, 14).Value = Sheets("sheet1").Cells(r - 1, 14).Value
 Sheets("sheet1").Cells(r, 15).Value = Sheets("sheet1").Cells(r - 1, 15).Value
 r = r + 1
End If
```

The address and coordinate search will be conducted using parcel numbers. The county treasurer's website maintains database entries for each tax category that is past due, so the same property can appear on multiple lines. Rather than performing a query on a parcel number for which the address and coordinates has already been gathered for, code was written to check if the current row it is evaluating has a duplicate entry and to copy that information from the duplicate entry into the current row. This code was written to avoid performing the same parcel query over and over again.

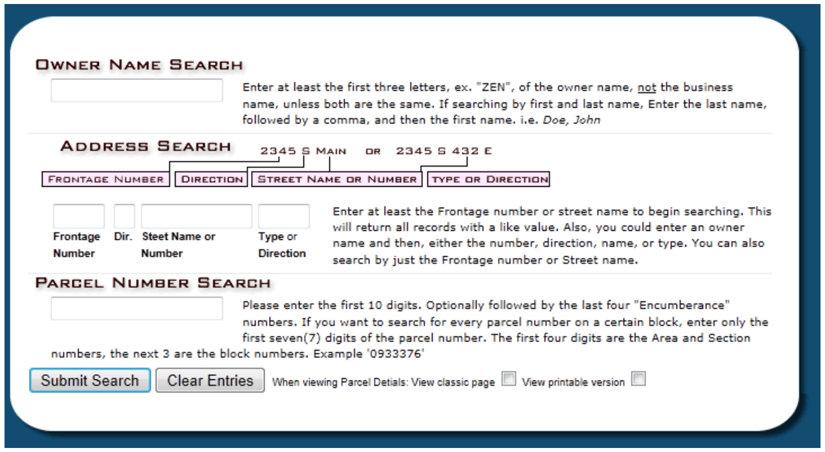
Accessing the County Assessor's Website

The following code was written to access the County Assessor's website:

```
a.visible = False
a.openpage "http://assessor.slco.org/cfml/query/query2.cfm", True
a.waitForLoad
a.document.getElementbyID("parcelid").Value = _
    Sheets("sheet1").Cells(r, 2).Value
a.followLinkByText ("Submit Search")
a.document.Forms(0).submit
a.waitForLoad
```

Because the time involved with this query is so extensive, I selected not to show internet explorer. This code is pretty straightforward and uses the class module "agent" prepared by Professor Allen.

This is what the website looks like. The parcel number search box is near the bottom of the site.


OWNER NAME SEARCH

Enter at least the first three letters, ex. "ZEN", of the owner name, not the business name, unless both are the same. If searching by first and last name, Enter the last name, followed by a comma, and then the first name. i.e. *Doe, John*

ADDRESS SEARCH 2345 S MAIN OR 2345 S 432 E

FRONTAGE NUMBER | DIRECTION | STREET NAME OR NUMBER | TYPE OR DIRECTION

Frontage Number | Dir. | Steet Name or Number | Type or Direction

Enter at least the Frontage number or street name to begin searching. This will return all records with a like value. Also, you could enter an owner name and then, either the number, direction, name, or type. You can also search by just the Frontage number or Street name.

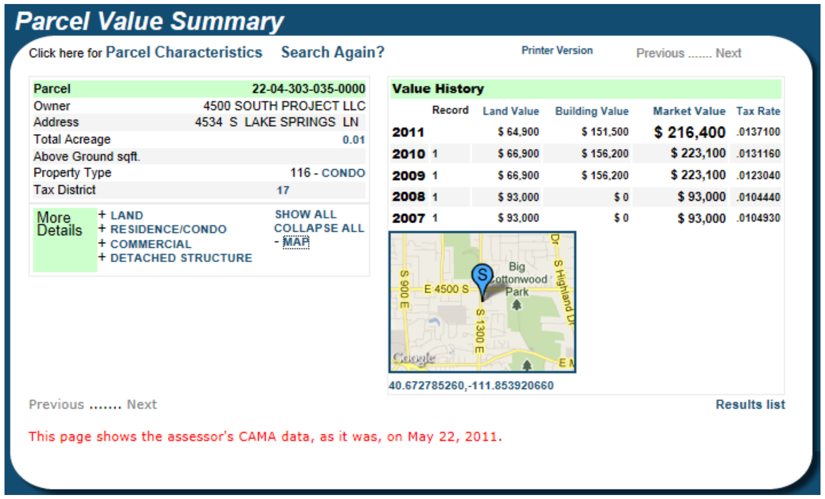
PARCEL NUMBER SEARCH

Please enter the first 10 digits. Optionally followed by the last four "Encumberance" numbers. If you want to search for every parcel number on a certain block, enter only the first seven(7) digits of the parcel number. The first four digits are the Area and Section numbers, the next 3 are the block numbers. Example '0933376'

When viewing Parcel Detials: View classic page ☐ View printable version ☐

Extracting Address Information

Once the parcel query is submitted the next website looks like this:



Parcel Value Summary

Click here for Parcel Characteristics   Search Again?   Printer Version   Previous .... Next

Parcel	22-04-303-035-0000
Owner	4500 SOUTH PROJECT LLC
Address	4534 S LAKE SPRINGS LN
Total Acreage	0.01
Above Ground sqft.	
Property Type	116 - CONDO
Tax District	17

More Details: + LAND, + RESIDENCE/CONDO, + COMMERCIAL, + DETACHED STRUCTURE   SHOW ALL   COLLAPSE ALL   - MAP

Value History

	Record	Land Value	Building Value	Market Value	Tax Rate
2011		$ 64,900	$ 151,500	$ 216,400	.0137100
2010	1	$ 66,900	$ 156,200	$ 223,100	.0131160
2009	1	$ 66,900	$ 156,200	$ 223,100	.0123040
2008	1	$ 93,000	$ 0	$ 93,000	.0104440
2007	1	$ 93,000	$ 0	$ 93,000	.0104930

40.672785260,-111.853920660

Previous .... Next   Results list

This page shows the assessor's CAMA data, as it was, on May 22, 2011.

The address information is located in the upper left and the coordinates appear directly below the Google Maps insert.

```
a.updateHTML 'reads the current html and puts it in the text property
 a.position = 1

   'the following extracts the address of the parcel and saves each element
   a.moveTo ("Owner")
   a.moveTo "style="
   address = a.getText("<")
   Debug.Print address
```

The first two lines of code above are copied from the "agent" file prepared by Professor Allen. The next lines of code navigate to the html source code with the text of "Owner" and then within that text section moves to the "style" tag and retrieves the text from the beginning of that tag to the symbol "<".

No Address Found

```
If InStr(1, address, "height:", vbTextCompare) > 0 Then
  Sheets("sheet1").Cells(r, 10).Value = "N/A"
  Sheets("sheet1").Cells(r, 11).Value = "N/A"
  Sheets("sheet1").Cells(r, 12).Value = "N/A"
  Sheets("sheet1").Cells(r, 13).Value = "N/A"
  Sheets("sheet1").Cells(r, 14).Value = "N/A"
  Sheets("sheet1").Cells(r, 15).Value = "N/A"
  GoTo Ending:
End If
```

Some parcel numbers do not have addresses associated with them. The code above enters N/A into each the corresponding cell for the address columns.

It then instructs the interpreter to go to the "Ending:" location of the sub-procedure which is near the bottom of the code, so that it avoids the code dealing with text manipulation, which I'll discuss next.

Address Text Manipulation

The address text that is returned from the html source code looks like this:

"float: right;">904&nbsp;&nbsp;E&nbsp;&nbsp;TENDOY&nbsp;&nbsp;CT&nbsp;&nbsp;

Code was written to find and save only the relevant portions of code, and saves each element into a different column. After processing this text, the result is:

Street Number: 904
Direction: E

Street Name: TENDOY
Street Type: CT

The code that accomplishes this text manipulation is below:

```
trimLength = InStr(1, address, ">", vbTextCompare)
address = Right(address, Len(address) - trimLength)
ampLoc = InStr(1, address, "&", vbTextCompare)
streetNum = Left(address, ampLoc - 1)
Sheets("sheet1").Cells(r, 10).Value = streetNum
direction = Mid(address, ampLoc + 12, 1)
Sheets("sheet1").Cells(r, 11).Value = direction
address = Right(address, Len(address) - (ampLoc + 24))
ampLoc = InStr(1, address, "&", vbTextCompare)
streetName = Left(address, ampLoc - 1)
Sheets("sheet1").Cells(r, 12).Value = streetName
address = Right(address, Len(address) - (ampLoc + 11))
address = Replace(address, "&nbsp;", "", 1, , vbTextCompare)
streetType = Trim(address)
Sheets("sheet1").Cells(r, 13).Value = streetType
```

Extracting Coordinate Information

The following code moves to the text where google maps is used and to the src= element. It returns the text from the end of this element title to the text "</a>". This text is in the format:

"http://maps.google.com/staticmap?center=40.672785260,-111.853920660&amp;zoom=13&amp;size=160x120&amp;maptype=roadmap&amp;markers=40.672785260,-111.853920660,blues&amp;key=ABQIAAAAPCYJ6hssEfoW91ymjvlIXhRYOFoJhxSZRZ7WSbHGx8xbol4G5xRcrRj2mSo2NzD-OpLj3cYhpbQdiA"><br>

40.672785260,-111.853920660"margin: 0px;" id="seek1" method="get" name="seek1" action="http://search.slco.org/search?">

The following code manipulates the text in order to extract the relevant information:

```
   'the following extracts the coordinates of the parcel and saves it
   a.moveTo "href=""http://maps"
   a.moveTo "src="
   coordinates = a.getText("</a>")
   trimLength = InStr(1, coordinates, "<br>", vbTextCompare)
   coordinates = Right(coordinates, Len(coordinates) - trimLength - 4)
   commaLoc = InStr(1, coordinates, ",", vbTextCompare)
   longitude = Left(coordinates, commaLoc - 1)
   lattitude = Right(coordinates, Len(coordinates) - commaLoc)

   Sheets("sheet1").Cells(r, 14).Value = lattitude
   Sheets("sheet1").Cells(r, 15).Value = longitude

Ending:

Next

End Sub
```

The result is:

Lattitude: 40.672785260
Longitude: -111.853920660

This information is then stored in sheet1. The location "Ending:" is referenced earlier in the sub-procedure and is navigated to in instances where a query has already been run for a duplicate entry (see above for where it is referenced). The sub-procedure is then ended.

Learning and Conceptual Difficulties

The most challenging aspect of this project was trying to find the appropriate code for extracting what I needed from the websites in question. Because the examples shared in class were in many ways not similar to the websites I was using, it was difficult to determine how to access the County Treasurer's website. As luck would have it, I figured it out, but only after tens of hours spent doing so. Once that was figured out, the rest of the assignment was not as challenging.

Had I more time to spend on the project, I would have added ribbon functionality, downloaded additional data, and formatted the file to make it more attractive. I also would have gone through my code to see how I could make it more efficient. I think spending fifty hours on a project should be enough, but these are just some thoughts for ways it can be improved.

As I side note, after spending considerable time trying to figure out how to access the County Treasurer's website, I almost gave up entirely and was about to change my final project (or not do it all and take a C in the class). It was a miracle that I figured it out and I'm glad I did. Now I have something to show to my friends and family for all the hard work!