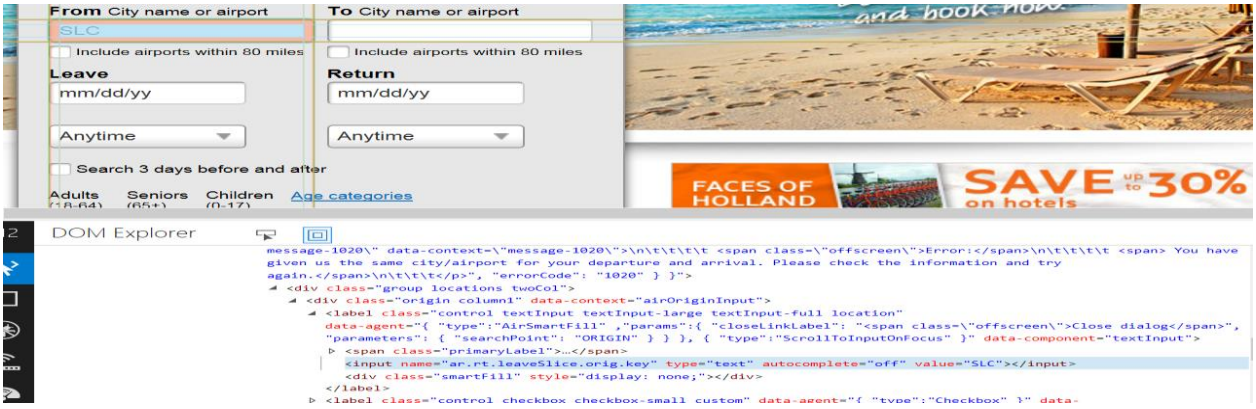
Executive Summary

My project's goal was the following, to have someone input information into the cells of the sheet that consisted of a departure city, a destination city, and the dates of their travel. Because of the numerous sites that claim to have the lowest price, most people have to check three or four places to verify that what they got was the cheapest possible flight. So to avoid this hassle, I created this VBA project that eliminates this need to check multiple places for flight information and gives you the lowest price possible for the flight.

The way that I have done this is by checking one of the most popular travel agencies in the world, Orbitz, and second checking Google.com/Flights which is simply a search feature that scans all the airlines personal websites and verifies through them what their flights are. This way I checked both a travel agency and with the airlines themselves to see who has the lowest price and then displayed the price to the customer along with a hyperlink to the lowest price's URL. In the following paragraphs

First Step: Understanding HTML and VBA

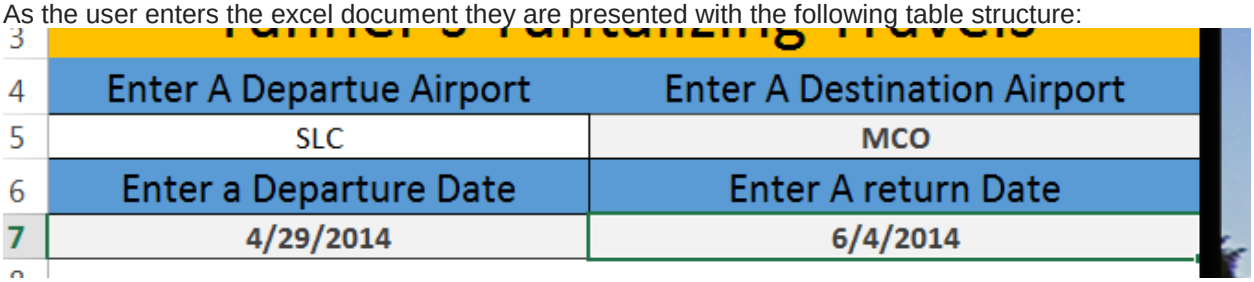
While my web programming skills are nothing short of embarrassing, I feel comfortable with reading what the browser is rendering to me so this was a good first step into solving the first problem of, how do I get information from the customer into the input boxes on the website?



In approaching this first question of how to input information into the designating fields that Orbitz gives, I needed to know the id's of the boxes or the names by which Orbitz identified them. So this involved me going with IE and using the inspect element tool to find the specific tag or div that these boxes were nested in and then getting there equivalent unique identifier to be used to locate them with VBA. This proved to be a little bit more challenging than originally thought because the structure of the webpage that Orbitz was using was a table structure with each cell being heavily formatted with multiple nested divs to structure it the way that they wanted. So I had to really dig to find at what depth the box was located and nothing else so that I prevented potentially changing other CSS or HTML in the page with my code.

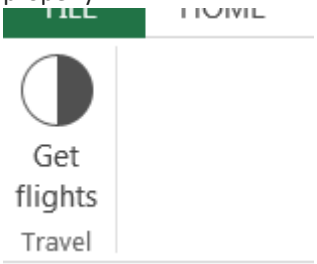
Second Step: Ribbon Click to start

As the user enters the excel document they are presented with the following table structure:



3		
4	Enter A Departue Airport	Enter A Destination Airport
5	SLC	MCO
6	Enter a Departure Date	Enter A return Date
7	4/29/2014	6/4/2014

The cells under the blue cells are the areas where the user would input the desired information. They can enter either airport code or City name. They need to enter dates in this format for the webpages to accept them properly.



Upon entering the information, the user then under the "Travel" tab is greeted by this button. Upon clicking it, the macro begins to run.

The sub below is code that is called when the Get flights button is clicked. It is at the top of all of my code to allow chronological progression of the code.

```
Dim cent As String
Dim lowPrice As String
Dim dollarS As String
Dim url As String
Dim hyper As Hyperlink
```

```
Sub All(control As IRibbonControl)
lowPrice = ""
'This is the Ribbion Click sub that calls the subs based on the length of strings
departure = Range("A5").Value
Destination = Range("D5").Value

    If Len(departure) < 4 And Len(Destination) < 4 Then
    Orbitz
    googleBackup
    Else: Orbitz
    End If
        If lowPrice = "" Then
        lowPrice = " There were no Flights available for the given set of inputs"
        End If

Range("A11").Clear
    Range("A11").Value = "$" & lowPrice
    'This sets the HyperLink to the Cell that the low price is set to

ActiveSheet.Hyperlinks.Add _
        Anchor:=Range("A11"), _
        Address:=url, _
        ScreenTip:="Travel"
End Sub
```

After setting the global variables that will be needed in multiple sub procedures or functions, the first lines set the departure and destination strings of the user input. The reason I do this here is to check whether the user entered a city name or an airport code. If either string is greater than 3 characters long, then I only run the Orbitiz code because only Orbitz handles city names. This is due from google not allowing people from automatically filling input fields with text and then executing a post call. I therefore had to edit the URL of google. This will be shown later in this report.

The other check that I do here is if there was a problem for some reason and no flights were entered or a webpage failed, I set the lowPrice string equal to the error message that tells the user that something went wrong and to try and run the process again. This lowPrice string gets edited by both the Orbitz and google sub procedures so if it was successful, then this error will not show.

Third Step: Calling Orbitz

Upon the All sub calling Orbitz, the Dims are set and then the Agent a is the first to have a url call.

```
price = 0
orbitzPrice = 0
a.visible = True
a.openpage "http://www.orbitz.com/flights/", True
```

The price and orbitzPrice are local variables that are used to verify that the lowPrice is really the low price from the webpage.

I left the visible to True so that the user would know that the system was working instead of them thinking that the click of the button did not register the action.

```
'Retreive Departure point from Excel sheet
deptPoint = Sheets("Sheet1").Range("A5")

'Retreive Departure Date from Excel sheet
DeptDate = Sheets("Sheet1").Range("A7")

'Retreive Arrival point from Excel sheet
retnPoint = Sheets("Sheet1").Range("D5")

'Retreive Arrival Date from Excel sheet
RetnDate = Sheets("Sheet1").Range("D7")

'Enter Departure Point in web page
Set departurepoint = a.document.getElementsByName("ar.rt.leaveSlice.orig.key")
departurepoint.Item(0).Value = deptPoint

'Enter Arrival Point in web page
Set ArrivalPoint = a.document.getElementsByName("ar.rt.leaveSlice.dest.key")
ArrivalPoint.Item(0).Value = retnPoint

'Enter Departure date in web page
Set DepDate = a.document.getElementsByName("ar.rt.leaveSlice.date")
DepDate.Item(0).Value = DeptDate

'Enter Return date in Web page
Set ReturnDate = a.document.getElementsByName("ar.rt.returnSlice.date")
ReturnDate.Item(0).Value = RetnDate
```

Next is the retrieving of information from the ranges that the user inputted. I then set those values to the designated input tags in the HTML so that webpage can run the search on the values. This is where the city name allows me to search for multiple airports with the city. If the user enters the full city name, then Orbitz searches all airports in the city and returns the lowest fare for the user. The other important attribute is the date format. Because I required the user to have the format previously mentioned, then the date can be accepted by the page.

```
'Once the information is inputed into the selected input boxes, then we click the search flight button to load the flights.
For x = 0 To a.document.All.Length - 1
    If LCase(a.document.All(x).tagname) = "input" Then

       If InStr(1, a.document.All(x).outerhtml, "<input name=""search"" type=""submit"" value=""Search Flights""") > 0 Then
          a.document.All(x).Click
       End If
    End If
  Next
a.waitForLoad
x = 0
```

After inputting the information, the code must select the “Search Flights” button on the webpage. To find the element of the search flight button the above code is run. It starts by looking through all of the HTML tags at each tag named “input”. Once it has that, it checks to see if within the tag’s html, it has the string in the if-statement. Only the search button had this string so I was guaranteed that if I found it, it was the one. Upon finding it, the agent then “Clicks” the button and it executes the call to the server to return the flight search based on the inputted information.

```
tag = "<div class=""airResultsCard withSelectSliceCheckbox optimizedCard withSelectFlightFindHotelLink"" data-context=""airResultsCard"">"
tag2 = "<span class=""money small-cents small-symbol"">"
```

These were the two tags that were needed to search for the returned price that was nested inside many other div and span classes.

```
'This is the logic for looking for the specific area of the webpage where the information of the flight is located. Iterates through the html looking
'for the specific tag.
For x = 0 To a.document.All.Length - 1
  If LCase(a.document.All(x).tagname) = "div" Then
    If Left(a.document.All(x).outerhtml, Len(tag)) = tag Then
      Set div1 = a.document.All(x)

      For y = 0 To div1.All.Length - 1
        If LCase(div1.All(y).tagname) = "span" Then
          If Left(div1.All(y + 1).outerhtml, Len(tag2)) = tag2 Then

              'once the the element is found, it then goes through the first price of the page and then ensures that no other prices can reset the low pric
               If price = 0 Then
               price = 1
               'the html is dirty and needs to be cleaned with the only numbers function
              orbitzPrice = OnlyNumbers(div1.All(y + 1).innerhtml)
               dollarS = Left(orbitzPrice, Len(orbitzPrice) - 2)
               cent = Right(orbitzPrice, 2)
               lowPrice = dollarS & "." & cent
               url = a.document.url

               End If

            Exit For
          End If
        End If
      Next
    End If
  End If

Next
```

The outermost for loop is searching through the entire HTML looking for all the div classes. If it is a div then it checks to see if it is the div that tag (our desired one) needs to be. This tag was the out box of the first result set on the page. The page is default to always show the lowest price first so I was safe in assuming that the first box will always be the lowest.
If it is the desired div that the code was looking for, then I set it to a div1 that is a dim object. Now instead of having to search the entire HTML I can look just inside this container that the other tag resides in instead of looking for it over the whole page.

It enters the y counter loop looking for tag2. Once the tag2 is found, then the price dim which was set to 0 at the beginning is checked to see if it is still zero. If it is, then the price is set to 1 which ensures the following code will only be run once. OrbitzPrice string is then passed to a function that takes the inner html and reduces it to only the numbers without all the HTML.

```
Function OnlyNumbers(ByVal WhichString As String) As Variant
    OnlyNumbers = CDbl(RegExpReplace(WhichString, _
                       "[^0-9]", vbNullString, True))
End Function
```

```
Function RegExpReplace(ByVal WhichString As String, _
                       ByVal Pattern As String, _
                       ByVal replaceWith As String, _
                       Optional ByVal IsGlobal As Boolean = True, _
                       Optional ByVal IsCaseSensitive As Boolean = True) As String
'Declaring the object
Dim objRegExp As Object
'Initializing an Instance
Set objRegExp = CreateObject("vbscript.regexp")
'Setting the Properties
objRegExp.Global = IsGlobal
objRegExp.Pattern = Pattern
objRegExp.ignoreCase = Not IsCaseSensitive
'Execute the Replace Method
RegExpReplace = objRegExp.replace(WhichString, replaceWith)

End Function
```

The above functions take a string and run a regex first and then replace method.

$342.00

So the regex looks for digits only and replaces anything that is not a digit with an empty string. It returns a value like 34200. This is why after calling the method I then do some string splitting that allows me to add the decimal back in and add the 00 after it. After setting the lowPrice equal to this new string, the URL from the search page is then stored in a global variable to later hyperlink the price back to this specific webpage.

The for-loop because it is nested will be rerun so that is why setting price = 1 is important to avoid all following prices will not affect it.

Orbitz is then finished processing.

Fourth Step: Calling Google

Once orbitz is finished and if the conditions are satisfied that it is only airport codes that is entered then the google sub procedure is called. For brevity I will skip the same processes that google has that are exactly the same as Orbitz.

```
DeptDate = Format(DeptDate, "yyyy-mm-dd")
RetnDate = Format(RetnDate, "yyyy-mm-dd")
b.visible = True
b.openpage "http://www.google.com/flights/#search;f=" & deptPoint & ";t=" & retnPoint & ";d=" & DeptDate & ";r=" & RetnDate, True
b.waitForLoad
tag = "<div class=""GLMPNSJCMLC""><div elt=""p""><div class=""GLMPNSJCLLC"">"
```

After declaring all the variables and then retrieving the information from the user, then the date must be formatted in the way google will accept it. This format is reverse of Orbitz and contains a 0 in front of the month and day if these values are single values. So the new format must be applied before the code alters the URL of the agent.

As you can see, these variables are inputted directly into the URL in the format that google accepts. Once this is done, the results are automatically rendered. No need to hit search buttons! The above tag is the particular div that google uses for their results.

```
'Iterate through and find the cheapest Price, then compare with Orbitz price and see who wins...
For x = 0 To b.document.All.Length - 1
    If LCase(b.document.All(x).tagname) = "div" Then
       If Left(b.document.All(x).outerhtml, Len(tag)) = tag Then

             If price = 0 Then
               googlePrice = OnlyNumbers(b.document.All(x).outerhtml)
               price = 1
               If Val(googlePrice) <= Val(lowPrice) Then
               lowPrice = googlePrice & "." & "00"
               url = b.document.url
               End If
           End If
           Exit For
       End If
    End If
  Next
```

Google's result were easier to get to because of the structure of the page. This allowed me to avoid nesting for loops and expedite the process. The difference between google and Orbitz in terms of how logic after the correct div is found is, I verify that the googlePrice is less than or equal to the lowPrice already set by Orbitz. If it is then the code runs very similar to Orbitz. The difference is in the cents portion of the price. Google does not show the cents for the fare so I hard code double 0 into the lowPrice if google is the low price. Always I set the URL to google if the prices are tied as well because of the ease of Google's display for the user and the direct access to the airline's website without getting involved with a travel agency. Finally, I exit the for-loop after this process to help speed up the process.

Google is then finished processing.

Fifth and Final Step: The End Result

The following image is the same as the fourth image shown under the second step of this paper

```
Dim cent As String
Dim lowPrice As String
Dim dollarS As String
Dim url As String
Dim hyper As Hyperlink

Sub All(control As IRibbonControl)
lowPrice = ""
'This is the Ribbion Click sub that calls the subs based on the length of strings
departure = Range("A5").Value
Destination = Range("D5").Value

    If Len(departure) < 4 And Len(Destination) < 4 Then
    Orbitz
    googleBackup
    Else: Orbitz
    End If
        If lowPrice = "" Then
        lowPrice = " There were no Flights available for the given set of inputs"
        End If

Range("A11").Clear
    Range("A11").Value = "$" & lowPrice
    'This sets the HyperLink to the Cell that the low price is set to

ActiveSheet.Hyperlinks.Add _
        Anchor:=Range("A11"), _
        Address:=url, _
        ScreenTip:="Travel"
End Sub
```

Once these methods return, the lowPrice is checked, the cell is cleared from previous searches, then if everything went according to plan then the lowPrice is set in the range. The Hyperlink is attached to the cell and it takes the user to the webpage where the lowPrice was located when clicked. After this final step, the final output looks like this :

	Tanner's Tantalizing Travels	
Enter A Departue Airport		Enter A Destination Airport
SLC		MCO
Enter a Departure Date		Enter A return Date
4/29/2014		6/4/2014
	TANNER FOUND YOU THIS FLIGHT	
$342.00		

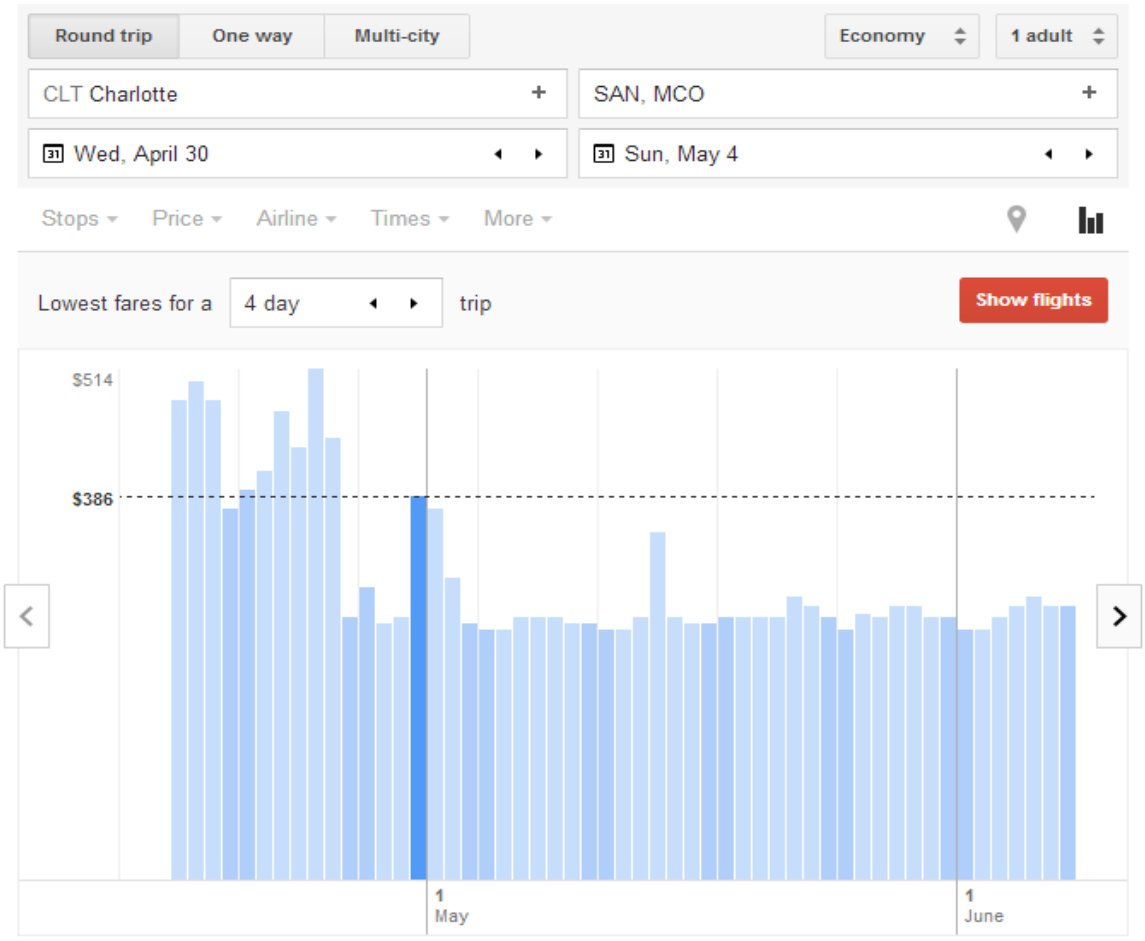
The Learnings and Difficulties

This project while I thought was fun, was extremely challenging at first to learn how to inerate through the specific divs to locate my desired information. Once I grasped these concepts, it was much easier and then I could add things like the Hyperlink and regex.

Over all it taught me what I desired to learn which was how to accurately glean information from other sources and collect this data into excel based off some parameters that the user inputs.

As always, VBA had her quirks and learning curves that I had to overcome. I still haven't gotten used to not having this be a MVL framework programming language. Also not having hints from the environment on how to solve errors or not having intellisense work with every object is hard to get used to.

Some things that I wish I could have done but couldn't were figuring out how to insert the values into Google to allow searching of city names as well as airport codes. Also I wish I could have implemented some sort of userform to take the data that returned the results is a cool display. Or I could have searched for flights on multiple days searching for the cheapest date to fly. The other thing is that google has a table view that is displayed after the results are returned which is just a graphical display of the results. I wish I would have had time to figure out how to import this graph to show the user how there price matches all the others. Below is a sample of the graph:



I did not receive any help from other students on this project. I did speak with Gove and his comment to me was “This is a worthy problem” when trying to help me with the div problem. So I felt justified in struggling with that.

All in all it was great.