

Get a Room!

Introduction

Each semester brings new group projects that require meeting together on many occasions. Usually the worst option is meeting at someone's house or apartment. Thus it falls on someone in the group (hopefully responsible) to remember to reserve a room for the group. However, if this person remembers too late, or even in time but during a busy week, then the chances of getting a room that suits the group's needs are very small. In fact, most of my group meetings occur after 5:00 PM due to this problem. Sometimes a group will find an open room during busy hours only to be kicked out at the next half hour because a different group had it reserved. When we learned how to access internet explorer, make JavaScript calls, and maneuver in an internet browser, I saw a solution to this problem. My project will involve entering in the times that our groups need to meet and four nights before the day my computer will automatically make the appropriate reservation.

Implementation

In an effort to practice writing good code and code that is relatively robust, the design of this program is laid out in many methods, each performing its unique function. The following table describes the class variables created and their description:

Name	Type	Description
ie	Object	This is the internet explorer instance and ultimately contains the html of the page and from which the code is executed and manipulated
HTML	String	A string copy of the html of the reservation page
dayTime	Type	This type is used to construct the JavaScript that is called from the code and has several variables associated with it.
startT	String	(of dayTime) The start time, retrieved from the spreadsheet and formatted appropriately.
endT	String	(of dayTime) The end time, retrieved from the spreadsheet and formatted appropriately.
room	String	(of dayTime) The room ID, assigned by the user from a list on Sheet 2.
block	String	(of dayTime) This is a variable used by the browser to know how many cells to color for the reservation. Each block represents 30 minutes, thus a two hour reservation will be 4 blocks. Ultimately this is the determinate of the length of time the reservation and also how many half-hour blocks of time the user may register in one day (the max being 4).
weekArray	Type	This type is used to retrieve information from the table of weekdays and associated preference reservation times entered by the user.
weekA()	weekArray	This is the array that holds the weekArray type objects.

automate	boolean	If the automation process is running, a true instance will run the "login" method without asking the user for authentication information but will retrieve it directly from the spreadsheet cells that ask for it.
----------	---------	--

The following table describes the methods used with their input parameters:

Name	Type	Parameters	Description
main	sub	None	This is the process for the manual entry room reserve. It calls other methods and functions to complete the process in an organized way.
Automated	sub	None	This is the process for the automated room reserve. It involves a lot of formatting and information retrieval from the spreadsheet. It also calls other methods.
login	sub	None	If the process is automated or manual this method handles how to authenticate onto the BYU Marriott School room reservation site.
manualReserve	sub	None	This method retrieves the data from the spreadsheet to formulate the JavaScript ajax call and makes the ajax call. It also handles formatting of the variables and is called by the manual entry process (the initial logic is organized differently for these processes, hence the need for a distinct method call than automation to send in the room request).
javaScript	Function	starttime, endtime, roomID, blocks	Receives the reservation inputs as parameters and makes the ajax call using the parameters passed through. This is the call to the BYU server.
openPage	Sub	Url as string	This opens the webpage, using the URL as an input parameter
openIE	sub	None	Creates an internet explorer object and instance and sets the browser to visible.
waitForLoad	sub	None	Allows the website to load without continuing on to the other processes
updateHTML	sub	None	This updates the HTML variable that stores the html for the site
saveFile	sub	path and theText	Saves the html to the path specified in the parameter

The “main” Sub Procedure

The “main” method is the process organizer for the manual entry process. This method is critical for calling the appropriate methods in the proper order for the manual entry of the reservation to be completed. The first thing this method does is request the user to input from the form “frmPassword”. Next the methods are called to open Internet Explorer and the login method called (see description below) to log the user in to the reservation site. The html of the destination site is saved to the same path as the workbook and then the reservation sub procedure

is called (see description below: manual reservation). When this sub procedure is done the entire process is complete, thus it both begins and ends the manual reservation method (manual meaning the date is chosen by the user).

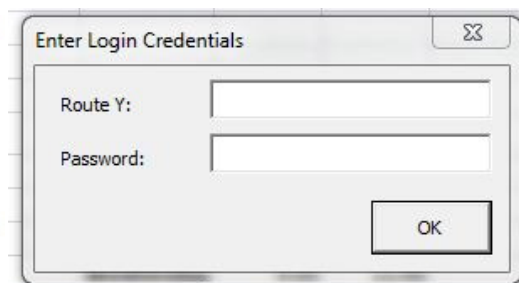
The “automated” Sub Procedure

This method cannot be called from the spreadsheet but is called from a VB script. Windows Task Scheduler runs this VB Script each night at 12:01 AM. The method is instructed to choose a date six days from the current date. This is significant because the browser only allows undergrad students to reserve rooms three days in advance, but grad students can reserve five days. At first I thought that bypassing the browser and calling the JavaScript directly allowed me to step over the “three day” security. But what actually happened was my status was change from undergrad to grad student. Thus I thought my testing was allowing me the extra days but in reality it wasn’t bypassing any security, a total let-down. This method creates an array with the preferred times of each day of the week that the user has chosen. The method then compares the date five days out to the day of the week in the array and chooses the appropriate time and creates the data string to be sent through JavaScript. It opens the website and presents the login screen. The login method is called, but because the user may not be at the computer when this runs, the login method gathers the authentication directly from the spreadsheet.

The “login” Sub Procedure

This method authenticates the user and puts that authentication information into the browser when it is called. This method also verifies whether the browser logged the user in automatically (by cookies) as often occurs. When I was testing the login ability, often the browser would go directly to the site without needing to authenticate, throwing several errors in my code. A for loop goes through the first 45 HTML tags and checks the innerText to see if my name and username show up (the username is in tag 34, so 45 should suffice any major changes to the code). If they do then the process is skipped. If they do not then the login process continues by inputting the form “frmPassword” information in the browser. The form screen shot and code for the “automated or manual” decision are pasted below.

```
'If the system is automating this, then don't
If (automate = False) Then
    'frmPassword.Show
    netID = frmPassword.txtRouteY
    password = frmPassword.txtPassword
    alreadyIn = False
Else
    netID = Reserve.Range("B15").Value
    password = Reserve.Range("B16").Value
    alreadyIn = False
End If
```



The code for verifying that the user isn’t already logged in is below:

```

'This checks to see if the browser already logged you in
For x = 1 To 45
    If ie.document.all.Item(x).innerText = "Mark Tuttle (sharktut)" Then
        Debug.Print "YES" & ie.document.all.Item(x).innerText
        alreadyIn = True
        Exit For
    End If
Next

'If the browser hasn't already logged you in, perform this action
If alreadyIn = False Then
    ie.document.forms("loginForm").all("net_id").Value = netID
    ie.document.forms("loginForm").all("password").Value = password
    ie.document.forms("loginForm").all("imageField").Click
End If

```

The “manualReserve” Sub Procedure

This procedure is called by “main” and is relatively simple. It was the last feature added to the code because it is valuable for a person who wants to reserve a time outside of the automated times and off-schedule for unforeseen needs of a room. It creates a type “daytime” object and sets the starttime, endtime, roomID, and block variables from the information provided in the spreadsheet. The JavaScript method is referenced to make the JavaScript call to the web server. The “Format” function formats the date and time how they need to be formatted when the ajax call is made.

```

Sub manualReserve()

Dim day As daytime
Dim c As String

c = Reserve.Range("A3").Value
day.startT = Format(c, "yyyy-mm-dd") & " " & Format(Reserve.Range("B3").Value, "h:mm")
day.endT = Format(c, "yyyy-mm-dd") & " " & Format(Reserve.Range("C3").Value, "h:mm")
day.room = Reserve.Range("E3").Value
day.block = Reserve.Range("D3").Value

ie.document.parentWindow.execscript (javascript(day.startT, day.endT, day.room, day.block))

```

The “JavaScript” Function

To get this information I had to analyze the JavaScript code that the web designers wrote. It was complicated to find which function did what and ultimately what needed to be done. I spent more than four hours searching both the JavaScript pages they have and the html to understand what was going on. Once I figured it out, I got help from Dr. Allen to learn how to call JavaScript from a VBA module, only to figure out that the very next day the web programmers introduced a completely new way of doing their site, using ajax. After much frustration I found again where they make the call to the web server and got more help from Dr. Allen. We recreated the ajax call and this is the result. It receives parameters and constructs the ajax. No matter what else, the code must appear exactly as it does when the JavaScript calls it. I attempted to make minor changes and it will not call until it is exactly the same. (See screen shot below).

```

Function javascript(starttime As String, endtime As String, RoomID As String, blocks As String) As String

javascript = " $.ajax({" & vbNewLine & _
    "type: ""POST""," & vbNewLine & _
    "url: ""/scheduler/reservation/reservation/""," & vbNewLine & _
    "data: ""startTime=" & starttime & "&endTime=" & endtime & "&roomId=" & RoomID & "&blocks=" & bloc
    "success: function(data, textStatus) {" & vbNewLine & _
        "    $("#resMessageText").text(data);" & vbNewLine & _
        "    $("#ReserveForm").dialog("close");" & vbNewLine & _
        "    $("#reservationMessage").dialog("open");" & vbNewLine & _
    "}" & vbNewLine & _
    "});"

End Function

```

Learning Details

Wireshark

This project presented several opportunities to learn additional utilities in order to perform this task. The first utility I used is an HTTP packet sniffer called Wireshark. I needed to use this tool because I needed to see what the parameter “event” looked like when the JavaScript made the web server call. Below is a screen shot of the packet that contained the string I needed and the url of the web server that I also needed.

6	0.612656	128.187.29.77	10.24.10.112	TCP	http > 54733 [ACK] Seq=1 Ack=1381 win=8704 Len=0
7	0.612765	10.24.10.112	128.187.29.77	HTTP	POST /scheduler/reservation/reservation/ HTTP/1.1
8	0.620975	128.187.29.77	10.24.10.112	TCP	http > 54733 [ACK] Seq=1 Ack=1601 win=11392 Len=0
9	1.427875	128.187.29.77	10.24.10.112	HTTP	HTTP/1.1 200 OK (text/html)

Frame 7 (274 bytes on wire, 274 bytes captured)

Ethernet II, Src: HonHaiPr_0c:e4:27 (00:19:7d:0c:e4:27), Dst: Cisco_3c:60:40 (00:11:bc:3c:60:40)

Internet Protocol, Src: 10.24.10.112 (10.24.10.112), Dst: 128.187.29.77 (128.187.29.77)

Transmission Control Protocol, Src Port: 54733 (54733), Dst Port: http (80), Seq: 1381, Ack: 1, Len: 220

[Reassembled TCP segments (1600 bytes): #5(1380), #7(220)]

Hypertext Transfer Protocol

Line-based text data: application/x-www-form-urlencoded

startTime=2010-03-18 19:30&endTime=2010-03-18 19:59&roomId=8&blocks=1&eventTitle=&eventDescription=

Highlighted in blue, line 7, is the line that contains the packet. In order to see these results, I looked for a packet that contained “POST” in the description. I later did a filter for the word “POST” and this was the first packet that contained it. On the very bottom of the screen shot there is a “Line-based text data:” section. When I expanded this line I saw the string that I needed (the very bottom “startTime=2010...”).

VB Script

In class we did one example of a VB Script and how it can be used to call Excel. I had never used VB Script before and was intimidated by it, not knowing how to do anything else besides open Excel. The class example code was great but wasn’t specific to the process I needed to run. I looked online at forums of how to run a VBA macro from VB Script. I finally found one and wrote the VB Script below to initiate the automated process. “Automated” is the sub procedure that is described above and called to run this process without any need for user input.

```

2  dim application
3      set application= createobject("Excel.Application")
4      application.Visible = true
5
6      application.workbooks.open "C:\Users\MarkTuttle\Documents\My Classe
7      application.Run "automated"
8      application.ActiveWorkbook.Save
9      application.ActiveWorkbook.Close
10     application.Quit
11     Set application = Nothing

```

Windows Task Scheduler

To schedule the automated process I searched how to run a batch file and put it in the registry as a program that runs when my computer is turned on. This was pretty complicated so I asked some friends if they had additional ideas. One recommended Windows Task Scheduler, which is relatively easy to use. The set up wizard allows for creating a basic task and below is a screen shot of browsing for my VB Script file.

Create a Basic Task

Trigger	Program/script:	
Daily	"C:\Users\MarkTuttle\Documents\My Classes\MISM 2\ISYS540 VBA\Pr Browse...	
Action	Add arguments (optional):	
Start a Program		
Finish	Start in (optional):	

JavaScript in VBA

Finding the appropriate JavaScript function was a challenge but I thought as soon as I found it I would be able to simply call it and reserve the room. It turned out to be a lot bigger challenge than that. Below is a screen shot of the original JavaScript file.


```

function makeReservation() {
    var startHour = Math.floor(startTime / 3600);
    var startMinute = (startTime % 3600 == 0)? '00': '30';
    var endVal = $("#endTime").val();
    var endHour = Math.floor((endVal - 60) / 3600);
    var endMinute = (endVal % 3600 == 0)? '59': '29';
    var start = ($("#start").val() + " " + startHour + ":" + startMinute); //MySQL does aut
    var end = ($("#start").val() + " " + endHour + ":" + endMinute);
    var blocks = ((endVal - startTime) / 1800);
    var eventTitle = ($("#eventTitle").val());
    var eventDescription = ($("#eventDescription").val());

    loadingContent("#reserveForm");

    $.ajax({
        type: "POST",
        url: "/scheduler/reservation/reservation/",
        data: "startTime=" + start + "&endTime=" + end + "&roomId=" + selectedRoomId +
        success: function(data, textStatus) {
            $("#resMessageText").text(data);
            $("#reserveForm").dialog("close");
            $("#reservationMessage").dialog("open");
        }
    });
}

```

Not being too familiar with JavaScript, it took me a long time to figure out what all the syntax meant. The variables all had different syntax, some having “\$” in front and others not being passed through or instantiated here. I went in for help from the professor who helped me recreate the JavaScript file in VBA. After two hours we were finally able to only call the “ajax” portion of the code without having to figure out where all those variable settings are originating. The function below is the result of what we worked on.

```

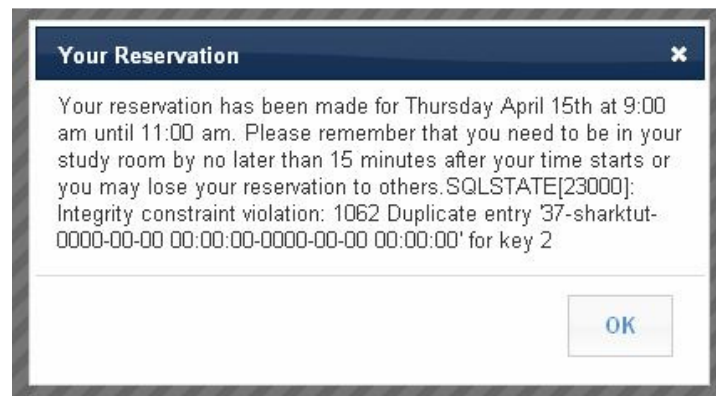
Function javascript(starttime As String, endtime As String, RoomID As String, blocks As String) As String
javascript = " $.ajax({" & vbNewLine & _
    "type: ""POST""," & vbNewLine & _
    "url: ""/scheduler/reservation/reservation/""," & vbNewLine & _
    "data: ""startTime=" & starttime & "&endTime=" & endtime & "&roomId=" & RoomID & "&blocks=" & bloc
    "success: function(data, textStatus) {" & vbNewLine & _
        "    $("#resMessageText").text(data);" & vbNewLine & _
        "    $("#ReserveForm").dialog("close");" & vbNewLine & _
        "    $("#reservationMessage").dialog("open");" & vbNewLine & _
    "}" & vbNewLine & _
    "});"
End Function

```

It mimics exactly what the original code calls and simply inserts the parameters it receives into the “data” section. One challenge that wasn’t foreseen was the very last line: “reservationMessage.dialog(“open”)”. This box was causing a verification of the reservation box to open that couldn’t be closed unless the user clicked “ok”. I attempted to change this line, add a line that closes the box (mimicking the line above it), and even deleting it all together. Any change to this ajax call caused the ajax to simply not run. I couldn’t find where this originated from besides just here. To get around this challenge, I had to redesign how the rooms were

scheduled from the spreadsheet so that only one reservation was made per day. This allows the browser to just close instead of having to click the box closed. Although this may seem like a less desirable result, the new design is actually better and more user-friendly.

A challenge that came about on my last test was not foreseen. On the reservation confirmation dialog box the standard text showed up with my reservation time with an SQL statement at the end. Below is a shot of the dialog box.



At first I thought this meant that I had logged in too many times but after some research it appears that there was a problem with the tables and a duplicate entry. The problem for my code is that this reservation wasn't actually recorded, even though the confirmation came through. I consulted with the professor who told me somewhere in my code I am making the same call twice. After reading my code I had a string variable that I was setting equal to the function. This was making the call once, and then the code quickly made the call again. Once I commented the first code out then the reservation worked just fine.

Arrays and Types

Although simple to most programmers I had to learn and use arrays efficiently in my code. I wanted to make an array of objects that held the day of the week and the hours that are to be reserved on each day. Here is the type that I created (left) and also the array of the type (right).

```
Type dayTime
    startT As String
    endT As String
    room As String
    block As String
End Type
Type weekArray
    dayName As String
    sTime As String
    eTime As String
End Type
Dim week() As dayTime
Dim weekA() As weekArray

ReDim weekA(6) As weekArray

    automate = True
    hoy = DateAdd("d", 6, Date)
    myDay = Format(hoy, "dddd")

'Builds the time and weekday array
For x = 0 To 5

    h = Reserve.Range("H" & x + 11).Value
    i = Format(Reserve.Range("I" & x + 11).Value, "h:mm")
    j = Format(Reserve.Range("J" & x + 11).Value, "h:mm")

    weekA(x).dayName = h
    weekA(x).sTime = i
    weekA(x).eTime = j

Next
```


Overall Experience

The project I chose is not the world's most challenging business problem, but it is a solution that, automated, is so much better for the user than manual entry. As such, I learned that business problems can be solved using VBA and other tools. These solutions simplify and improve the life of the business. This can also provide advantages to businesses. For example, I now have an advantage on reserving rooms over *all other students* because I don't have to "remember" each day when the day becomes available. Rather my code is run automatically. This will allow me prime access to the best rooms that other students will not have. If this were a business it would provide a competitive advantage that would allow a company to grow better and faster than competitors.

I also learned the value of choosing something unfamiliar and learning how to pursue solutions to problems that are unclear or vague. I had not really used an HTTP sniffer before, had never written anything in VB Script, nor used Windows Task Scheduler. It can be intimidating learning new tools, but I learned that with some study, desire, and hard work most any tool can be learned and utilized to solve the problem.

It was very helpful for me to choose my own project because I was forced to think through things on my own and come up with a solution that worked. It was also helpful because it was something I was interested in and was personally invested in finding the solution to. Since starting this project I have been able to implement these skills to help a TA write code to email all of the students of the class the scores on their tests, saving him four hours after each test. All of this allowed me to learn the value of automation and taking responsibility over a project and will hopefully lead to lucrative opportunities in the future.