

## FINAL PROJECT WRITE UP

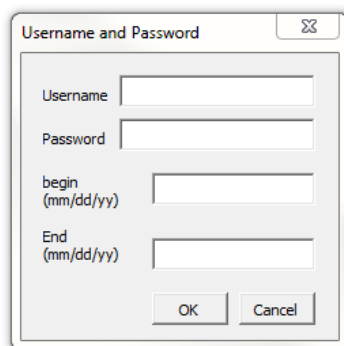
### 1. Executive Summary

This project is intended to help to organize the finances of the “programmer” who has checking and savings accounts in the Wells Fargo Bank and a credit card account with American Express. It allows the user to enter his username and password, define the time range desired and then download the statements and organize it month by month including totals. The program will allow the user to do in a couple minutes what has been done in about 30-45 minutes and will facilitate all sorts of analysis he will be able to do with the data arranged in an organized and neat way. This will make easier for him to track his financial behavior and to better make financial decisions for the future.

### 2. Implementation

#### a. Downloading the data

The first thing necessary to implement the project is to access the account information in the Wells Fargo webpage and the American Express account. To do so and to not have the username and password hardcoded, I decided to use a user form to capture this information through user input. The user form looks this:



Username is stored as txtUsername

Password is stored as txtPassword

Begin date is stored as txtBegin

End date is stored as txtEnd

Before proceeding, the program needs to check if all fields are filled and since the username and password can be any kind of string, it will not check any specific format for that. The begin date and end date need to follow a specific format (mm/dd/yy) so the program will check for the length of the string (8 characters) and for the presence of the slashes in the right place.

```
If txtUsername.Text = "" Or txtPassword.Text = "" Then
```

```

        MsgBox "Username and password required."
        Exit Sub
    ElseIf txtBegin.Text = "" Then
        MsgBox "Date range required."
        Exit Sub
    ElseIf Len(txtBegin) <> 8 Or Len(txtEnd) <> 8 Then
        MsgBox "Please make sure you enter the dates in the format mm/dd/aa"
        Exit Sub
    ElseIf Left(txtBegin, 2) > 12 Then
        MsgBox "Please make sure you enter the dates in the format mm/dd/aa"
        Exit Sub
    ElseIf Mid(txtBegin, 6, 1) <> "/" Then
        MsgBox "Please make sure you enter the dates in the format mm/dd/aa"
        Exit Sub
    (...) → repeat basically the same commands for txtEnd
End If

```

After checking the parameters, the program opens the Internet Explorer web browser and access the Wells Fargo login page.



Entering the data into the page was basically done in two different ways throughout this project. All of them used the code developed by Professor Allen called “agent”, which had to be previously imported into the excel file.

The first way to enter data in a specific field in the html code is to look for the “id” of the desired field and then determine its value. For example, to enter the user name in the page above, I simply looked for the “id” of the “username” field, which in this case is “userid”, then I set its value to be equal the user name previously typed in the user form and stored in a variable called “userNameWF”. Similar to that was also the method used to click the sign on button, which “id” is “btnSignon”.

```

agent1.explorer.Document.All("userid").value = userNameWF
agent1.explorer.Document.All("btnSignon").Click

```

The second way is to look for a group of characters, or string that would be right before the wanted link and use the “moveto” method to move the insertion point to the desired position, which is right before the

begin of the wanted *url* address. The program then stores the url in a variable called “link” and uses the method “open page” as in the example below:

To access the link to the account “Complete Advangate (RM)XXXXXX2593” below the following code was used:

Cash Accounts	
Account	
<a href="#">COMPLETE ADVANTAGE(RM)XXXXXX2593</a>	
<a href="#">SAVINGS XXXXXX4723</a>	
Total	

```

agent1.updateHTML
agent1.position = 1
If Not agent1.moveTo("COMPLETE ADVANTAGE") Then
    MsgBox "String not found in page's source 1"
    Exit Sub
End If
agent1.moveTo "href="""
link = agent1.getText("""")
agent1.openpage link
agent1.waitForLoad

```

Using either of the aforementioned approaches, it was possible to access all the necessary pages within the Wells Fargo webpage, select checking and savings accounts, enter the desired data range (previously typed in the user form) and finally download the statements in the “.csv” format.

#### Step 1: Choose an account & click Select.<sup>1</sup>

Account	COMPLETE ADVANTAGE(RM) XXXXXX2593	Select
---------	-----------------------------------	--------

#### Step 2: Verify the pre-filled date range.<sup>2</sup>

Date range	<table> <tr> <td>From</td> <td>To</td> </tr> <tr> <td>09/13/10</td> <td>12/10/10</td> </tr> <tr> <td>(MM/DD/YY)</td> <td>(MM/DD/YY)</td> </tr> </table>	From	To	09/13/10	12/10/10	(MM/DD/YY)	(MM/DD/YY)	<p>Extended History Now Availal</p> <p>To download more than 90 d you selected a date range gr activity page.</p> <p><i>Note: Always confirm "From" account activity.</i></p>
From	To							
09/13/10	12/10/10							
(MM/DD/YY)	(MM/DD/YY)							

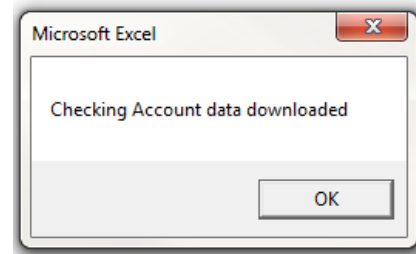
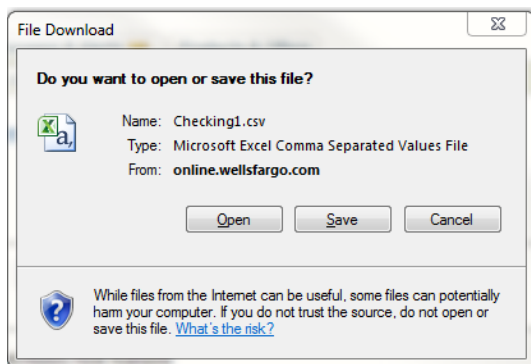
#### Step 3: Select a file format to download.<sup>3</sup>

File Format	<input type="radio"/> Quicken® 2005 (MAC) & 2008 (PC) or later ( <a href="#">Web Connect</a> ) <input checked="" type="radio"/> Comma Delimited (ASCII Spreadsheet)
-------------	--

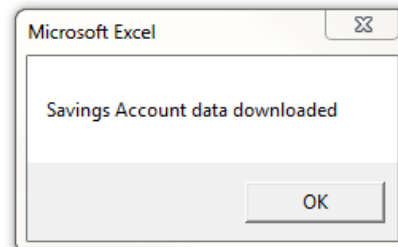
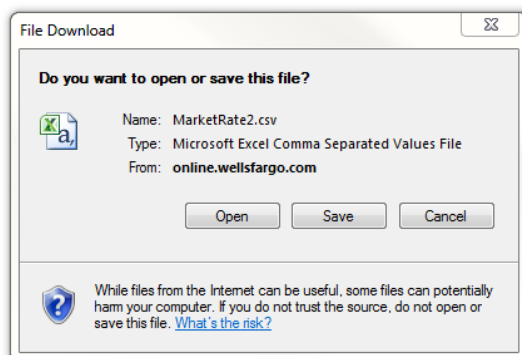
Download

As per the download, the user has to interact at this point and choose the location to save the “.csv” file. For the purpose of this code, the location must be the desktop.

When the first file is downloaded the program will show a message box warning that the file has been downloaded successfully.



After clicking “ok” the program will go through the same process to download the Savings account information and the following boxes will be seen.



At this point, the program should go to the American Express web page and do a similar process to download the information about transactions selecting the option “year to date” for the time period. Unfortunately the “agent” is not being able to interact with Amex’s website and at this point the user will have to download the file manually.

## b. Importing data to the spreadsheet

Below is a snapshot of how the spreadsheet looks like (the characters have been changed for privacy reasons). It has 12 worksheets (one for each month) and stores the data pulled from the statements previously downloaded. The code sorts the transaction by date and allocates them into the respective worksheets. To do so, the user just needs to click on the red button “Process Data”.

	A	B	C	D	E	F	G	H
1	DATE	DESCRIPTION	VALUE					
2	01/04/10	ΧΗΕΧΚ ΧΡΔ ΠΥΡΧΗΑΣΕ 01/02 ΚΡΙΣΙΤΨ ΚΡΕΜΕ ΔΟΝΥΤΣ ΟΡΕΜ ΥΤ 486830	\$ (12.76)					
3	01/04/10	ΠΟΣ ΠΥΡΧΗΑΣΕ – ΩΑΛ–ΜΑΡΤ #1768 ΩΑΛ–ΜΑΡΤ #ΟΡΕΜ ΥΤ 9418	\$ (67.59)					
4	01/04/10	ΩΠΗΔΑΩΑΛ ΜΑΔΕ ΙΝ Α ΒΡΑΝΧΗ/ΣΤΟΡΕ	\$ (700.00)					
5	01/04/10	ΒΨΥ ΒΟΟΚΣΤΟΡΕ 293020ΠΡΟΞΟ	\$ (123.08)					
6	01/06/10	ΒΨΥ ΒΟΟΚΣΤΟΡΕ 293020ΠΡΟΞΟ	\$ (27.90)					
7	01/06/10	ΒΨΥ ΗΟΥΣΙΝΤ & ΔΙΝΙΝΤΙΠΡΟΞΟ	\$ (715.00)					
8	01/07/10	ΩΕΝΔΨΣ #4710 ΘΠΡΟΞΟ	\$ (3.22)					
9	01/08/10	Νετφλίξ Λοσ Γατος	\$ (9.61)					
10	01/09/10	ΣΝΑΠΦΙΣΗ ΣΝΑΠΦΙΣΗ ΣΑΝ ΦΡΑΝΣΙΣΧΟ	\$ (2.46)					
25	TOTAL		\$ 256.59					

Obtain financial data  
WF Checking and Savings

Process Data  
Wells Fargo and Amex

The first thing the code does is to clear the current content of the worksheets (since it has accumulated transactions since the beginning of the year. Below is the code lines used to do so.

```

For monthNumber = 1 To 12
    Windows("Final Project.xlsm").Activate
    Sheets(monthNumber).Activate
    Range("a2").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.Clear
    Range("a2").Select
Next

```

After cleaning the worksheets, the program opens the three downloaded files, eliminates the unnecessary columns and organizes the files to bring them to the same format. Below is a sample of a line of code to open one of the files.

```
Workbooks.Open Filename:="C:\Users\Marcelo\Desktop\MarketRate2.csv"
```

Since the credit card payment is done through a withdrawal from the checking account, the code needs to eliminate the payment information; otherwise it would offset the expenses recorded in the credit card statement and give a wrong result. This task is performed by the following lines of code:

```

rowNumber = ActiveCell.Row
lastRow = Cells(rowNumber, 1).End(xlDown).Row
For rowNumber = rowNumber To lastRow
    On Error GoTo errorHandling
    Cells.find(What:="AMERICAN EXPRESS", After:=ActiveCell, LookIn:= _

```

```

        xlFormulas, LookAt:=xlPart, SearchOrder:=xlByColumns, SearchDirection:= _
        xlNext, MatchCase:=False, SearchFormat:=False).Activate
    ActiveCell.Rows("1:1").EntireRow.Select
    Selection.Delete Shift:=xlUp
Next
errorHandling:
Range("a1").Activate

```

Similar operation is performed in the Amex downloaded statement to delete the lines that record the payment of the debt because; again, it would offset the expenses and the user would end up with the wrong result.

The next step is to bring all transactions information to one file (I choose to put everything in the file downloaded from the checking account). Afterwards the transactions are sorted by date.

At this point the program cuts and pastes the data, month by month, into the respective worksheets in the original workbook. This is performed by the following lines of code:

```

For monthNumber = 1 To lastMonth
    Windows("Checking1.csv").Activate
    Sheets("Checking1").Activate
    Range("d:d").Select
    Cells.find(What:=monthNumber + 1, After:=ActiveCell, LookIn:= _
        xlFormulas, LookAt:=xlPart, SearchOrder:=xlByColumns, SearchDirection:= _
        xlNext, MatchCase:=False, SearchFormat:=False).Activate
    endRow = ActiveCell.Row - 1
    ActiveCell.Offset(-1, 0).Select
    Range(Selection, Selection.End(xlToLeft)).Select
    Range(Selection, Selection.End(xlUp)).Select
    Selection.Cut
    startRow = endRow + 1
    Windows("Final Project.xlsm").Activate
    Sheets(monthNumber).Activate
    Range("a2").Activate
    ActiveSheet.Paste
Next

```

Then some formatting is done to give the final appearance showed above.

Finally, the program will close the downloaded files and delete them from the desktop

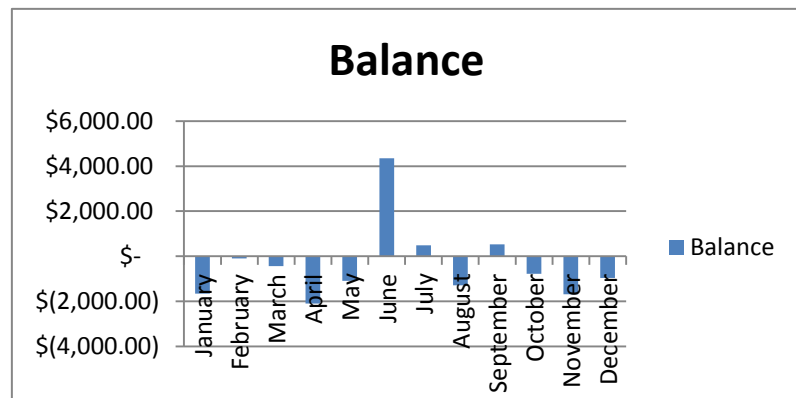
```

Windows("Checking1.csv").Close False
Windows("MarketRate2.csv").Close False
Windows("Summary.xls").Close False
Kill ("C:\Users\Marcelo\Desktop\Checking1.csv")
Kill ("C:\Users\Marcelo\Desktop\MarketRate2.csv")
Kill ("C:\Users\Marcelo\Desktop\Summary.xls")

```

With the final result, the user is able to do different kinds of analysis using an neatly organized workbook that summarizes all of his/her financial operations. As an example, the summary worksheet shows the balance of each month and shows in red those when expenses were greater than income. The user can do all sorts of analysis through graphics and identify the causes of deficits and superavits in his/her finances.

	A	B
1	<b>SUMMARY</b>	
2		
3	<b>Month</b> ▼	<b>Balance</b> ▼
4	January	\$ 256.59
5	February	\$ (97.99)
6	March	\$ (436.87)
7	April	\$ (2,099.05)
8	May	\$ (1,096.50)
9	June	\$ 4,353.38
10	July	\$ 480.61
11	August	\$ (1,288.15)
12	September	\$ 531.41
13	October	\$ (782.14)
14	November	\$ (92.91)
15	December	\$ (958.38)
16		
17	<b>Balance of Year to date</b>	<b>\$ (1,230.00)</b>



To keep this file updated had been a pain to the user and requires a fair amount of time and exposes him to failures, repetition of values etc. Having this process automatized makes it to take just a couple of minutes to go through the whole process with no risk of errors.

### 3. Learning and Conceptual Difficulties

Working with HTML programing language was an extra challenge because it is something completely new for me and it is very complicated because the html page is dynamic and referring to specific points in the text is somewhat dangerous because any change in the components before it would make the VBA code lose the reference to it. Because of that it is necessary when always that it is possible to use relative references.

Because of time constraints I couldn't resolve the problem with accessing the American Express page, so the download of that file is, for now, accomplished manually. Apparently the "agent" needs to be changed somewhat to resolve this problem, what I plan to do soon allowing the process to become entirely automated.