12/9/2010

Using Mint.com to Creating Forward-Looking Personal Account Budgets Based on Historical Purchase History

Dustin Schick
MBA 612

Executive Summary

Mint.com is a great personal finance website that aggregates all of your financial accounts and all of their transactions in one convenient place. These accounts include high yield money market accounts, checking and savings accounts, credit card accounts, student and home loan balances, IRA and investment accounts, estimated property balances, and more. The website is easy and convenient to use, and requires management of only one login and password to get a snapshot of your personal assets and liabilities as opposed to checking each account individually.

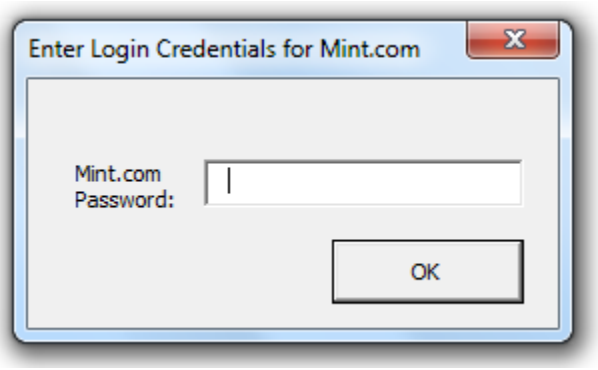
Mint.com also has a budgeting tool so you can set budgets for each of your estimated expenses. When establishing budgets, a user is on his own to guess where to set those budgets. For fixed costs such as monthly rent, car payments, auto insurance, etc., budgeting is usually the same every month. However, for more variable costs such as groceries, takeout, utilities, shopping, etc., creating a proper budget is more difficult. To make my forward-looking budgets easier to establish, I have created a program that pulls off my mint.com account balance and transaction data, then totals average purchase amounts in each of the 14 expense accounts that I use most frequently (but that have the most variability from period to period). These expenses are then averaged based on the current quarter of the year, the last quarter of the year, and monthly averages for year-to-date expenses. A proposed monthly budget amount is then generated for each of these variable expense accounts based on historical purchases. This gives me a very good guideline of realistic budgets that I can set for my finances.

To manipulate each of these expense accounts manually easily takes 30 minutes to an hour on every occasion, and would require results to be incrementally recorded…a big hassle. This program operates with the ease of a click of the button, inputting my mint.com password, and 30 seconds of computing.

Project Description and Implementation

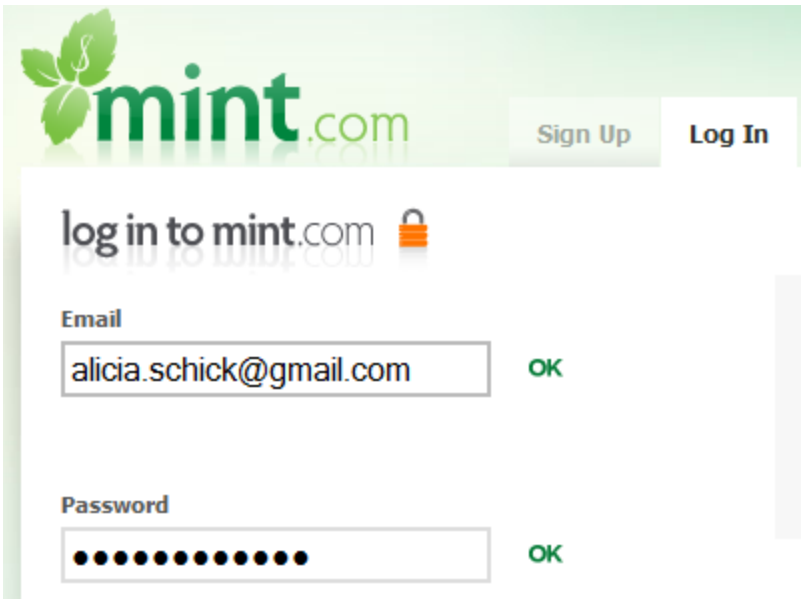
Mint.com is great at aggregating my account totals and transactions, but I wanted to create an easy way to aid me in establishing realistic budgets in each of my *variable* purchase accounts based on my historical buying behavior.

To do this, I relied heavily on Gove Allen's "agent" class module to interact with the mint.com website to assist in username and password login and input, plus downloading and interacting with the site. I combined this with a password form to hide my password, requiring it to be dynamically input each time without being saved in the file for anyone to see.



Using the agent and password form, I was able to login to mint.com through excel to extract all of my up-to-date account totals and complete transaction histories. One of the biggest challenges of this was working out the bugs to get this to function properly. For example, it took me a very long time to learn that my Internet Explorer browser's settings had to have automatic prompting for activex controls and automatic prompting for downloads enabled.

I still think it is absolutely incredible that I can automate the process of logging into a secure website using excel – that is awesome!



Below is the code to login and download the account balance information:

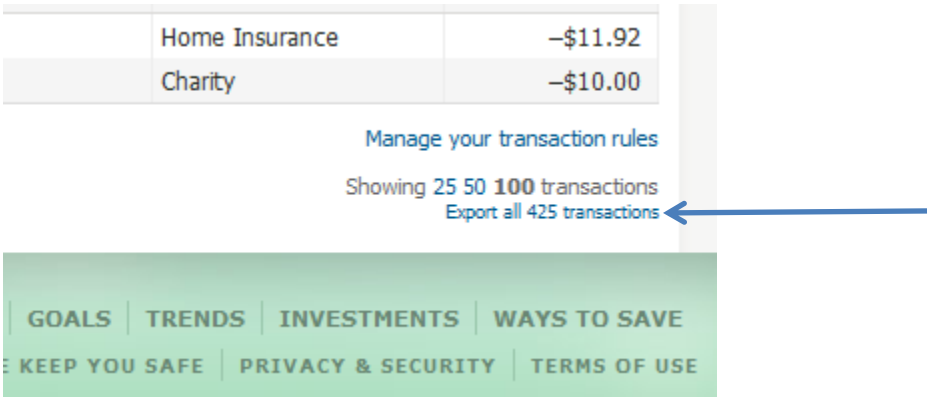
```
Sub login()
frmPassword.Show 'displays the form for the user to input the Mint.com password
agent1.visible = True
agent1.openpage "https://wwws.mint.com/overview.event"
agent1.explorer.document.all("form-login-username").Value = "alicia.schick@gmail.com"
Application.DisplayAlerts = False
Windows("Mint v2.0.xlsm").Activate
Sheets("Account Balances").Delete
Application.DisplayAlerts = True
agent1.explorer.document.all("form-login-password").Value = frmPassword.txtPassword.Text
SendKeys "{ENTER}"
agent1.waitForLoad
agent1.importPage "Account Balances"
Application.Wait Now + TimeValue("00:00:02")
End Sub
```

I then created a separate subprocedure to download my complete transaction history and close the internet explorer browser, as seen in the following code:

```
Sub downloadTransactions()
agent1.openpage "https://wwws.mint.com/transaction.event?"
agent1.waitForLoad
agent1.explorer.document.all("transactionExport").Click
```

```
Application.Wait Now + TimeValue("00:00:02")
SendKeys "o"
agent1.terminateIE
End Sub
```

At first, it was quite tricky to figure out how to download all of my transactions, but with the help of Internet Explorer's developer tools, I was able to figure out the ID of the total transactions export area that I wanted to interact with. I could not figure out any way other than using SendKeys to interact with the IE's browser to export the data. SendKeys works as long as I give it the undivided opportunity to interact with the browser.



By far the greatest difficulty I had was transporting the transactions data from the temporary download file into the excel workbook that the program is based in. Unfortunately, I spent close to 10 hours on this one little piece alone, mostly troubleshooting, talking with people, and seeing error after error…! I finally got this functionality to work using the following code:

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

This gave the time buffer that was crucial to the operation (which was finally what enabled the program to work propertly), coupled with this code:

```
Sub CopyPasteTransactions()
Application.DisplayAlerts = False
Sheets(Worksheets.Count).Select
Sheets(Worksheets.Count).Move After:=Workbooks("Mint v2.0.xlsm").Sheets("Command Center")
```

```
ActiveSheet.Select
ActiveSheet.Name = "transactions"
Application.DisplayAlerts = True
ActiveSheet.Previous.Select
Range("A1").Select
```

Once all of the data was actually inside my workbook, the programming was much more intuitive. I have an “Account Balances” tab with all of my total account aggregates and a separate “transactions” tab with historical data of every single transaction: when, where, what amount, which account, etc. I created additional tabs with dynamic pivot tables that parsed the data into expense accounts according to spending in this quarter, last quarter, and year to date.

From here, I used the VBA macro recorder to search out each expense account and put its quarterly and monthly year-to-date averages into the main control panel of the spreadsheet. Lastly, I dynamically created proposed monthly budget amounts based on averages from the last two quarters. This is a great aid in establishing and tweaking budget amounts in each category based on my personal purchasing habits!

All of this is executed by one click of the “Download Mint Data” button!

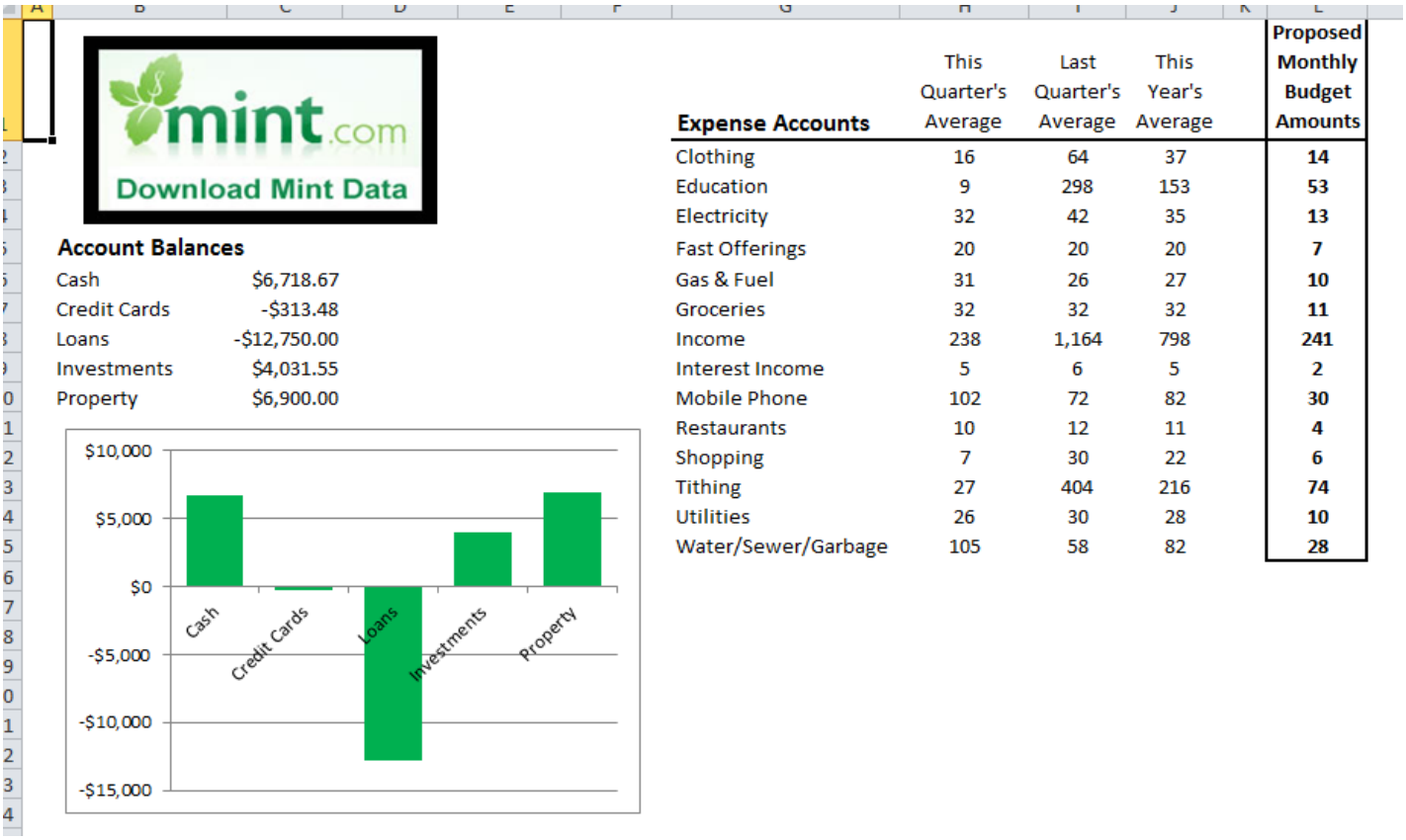
Total lines of code: 598

Learning Points

- The login and protected password functionality were very difficult and took far longer than I ever thought. With that being said, however, I feel confident that I could do this for another website and application very quickly now
- Discovering how to find mint.com’s website field IDs was a huge “aha” moment. Before making this discovery, I had been trying to download separately each page of historical account transactions (100 transactions each) and was having all sorts of problems.

- After downloading these transactions, they were placed in a temporary workbook. I thought the process would be simple and painless to move this worksheet into the program's workbook, but I spent by far more time on this single piece than anything else. I asked for help from several different people and each time I thought I figured out the problem, but each time (except the last, of course) it wasn't quite enough. However, this taught me that there are ways to get around difficult challenges and it is important to both troubleshoot incrementally and reach out for help from others.
- I am amazed at the steep learning curve I climbed to make this project work, and I am excited to continue learning and using VBA to automate even more tasks in the future! As I continue the long journal of VBA mastery, I see very valuable applications of this tool to help set me apart in my future professional experiences!

Lastly, here is a screenshot of my forward-looking budgeting tool (based on historical purchases):



Expense Accounts	This Quarter's Average	Last Quarter's Average	This Year's Average	Proposed Monthly Budget Amounts
Clothing	16	64	37	**14**
Education	9	298	153	**53**
Electricity	32	42	35	**13**
Fast Offerings	20	20	20	**7**
Gas & Fuel	31	26	27	**10**
Groceries	32	32	32	**11**
Income	238	1,164	798	**241**
Interest Income	5	6	5	**2**
Mobile Phone	102	72	82	**30**
Restaurants	10	12	11	**4**
Shopping	7	30	22	**6**
Tithing	27	404	216	**74**
Utilities	26	30	28	**10**
Water/Sewer/Garbage	105	58	82	**28**