EXECUTIVE SUMMARY

I have long been an avid user of Pandora.com's internet radio website. Pandora works by taking songs and/or artists provided by the user, identifying specific traits in the songs (or songs by the provided artist), and then looking for songs with similar traits from a vast library of music. As an example of the traits identified by Pandora, the following is a list of traits for the song "Flake" by Jack Johnson:

- mellow rock instrumentation
- a subtle use of vocal harmony
- mild rhythmic syncopation
- acoustic sonority
- major key tonality
- a vocal-centric aesthetic
- a dynamic male vocalist
- slide/pedal steel guitars
- acoustic rhythm guitars

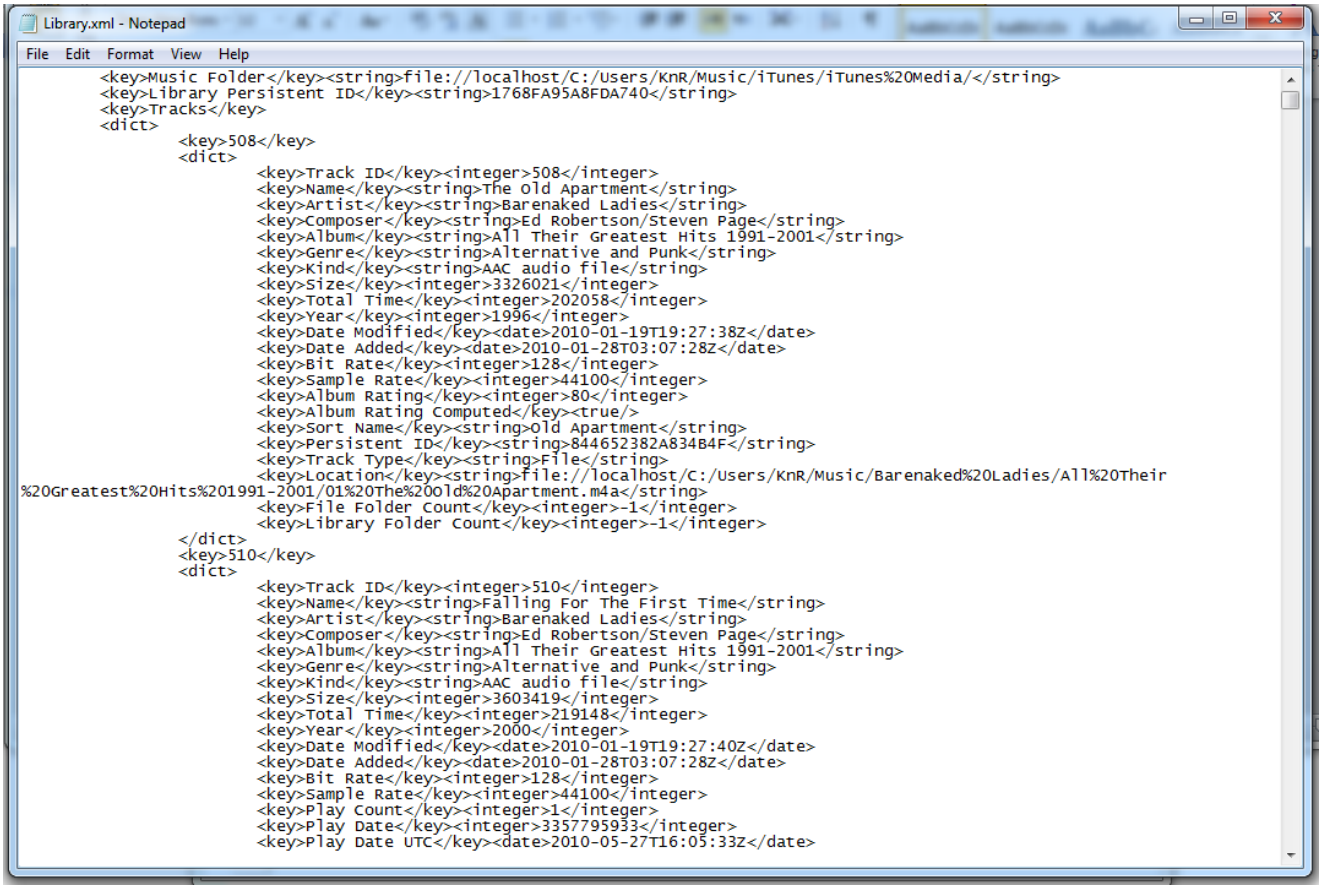
My project aims to draw upon Pandora's collection of traits and songs to create an individualized "station" (that is, a list of songs) from the songs in the user's iTunes library. Specifically, the program will read the songs in an XML file (the file type exported by iTunes) and allow the user to select a song or songs; the program will then identify the traits in those selected songs. The user can then select any number of these traits and, at the push of a button, the program will find ALL songs in the user's iTunes library which feature those selected traits. In the end, much like a Pandora radio station, the user will have displayed a list of songs that boast a particular desired musical trait. Although Pandora naturally does not have data for all songs, Pandora's collection of traits is still impressive and should make for an interesting method of finding songs similar to those chosen by the user.

Important Note: For a library with hundreds/thousands of songs, finding traits for each song online takes a LONG time. The workbook posted here has most of the songs saved in an archive; only a few songs will be searched for online for the sake of time.

IMPLEMENTATION

Loading Songs from iTunes

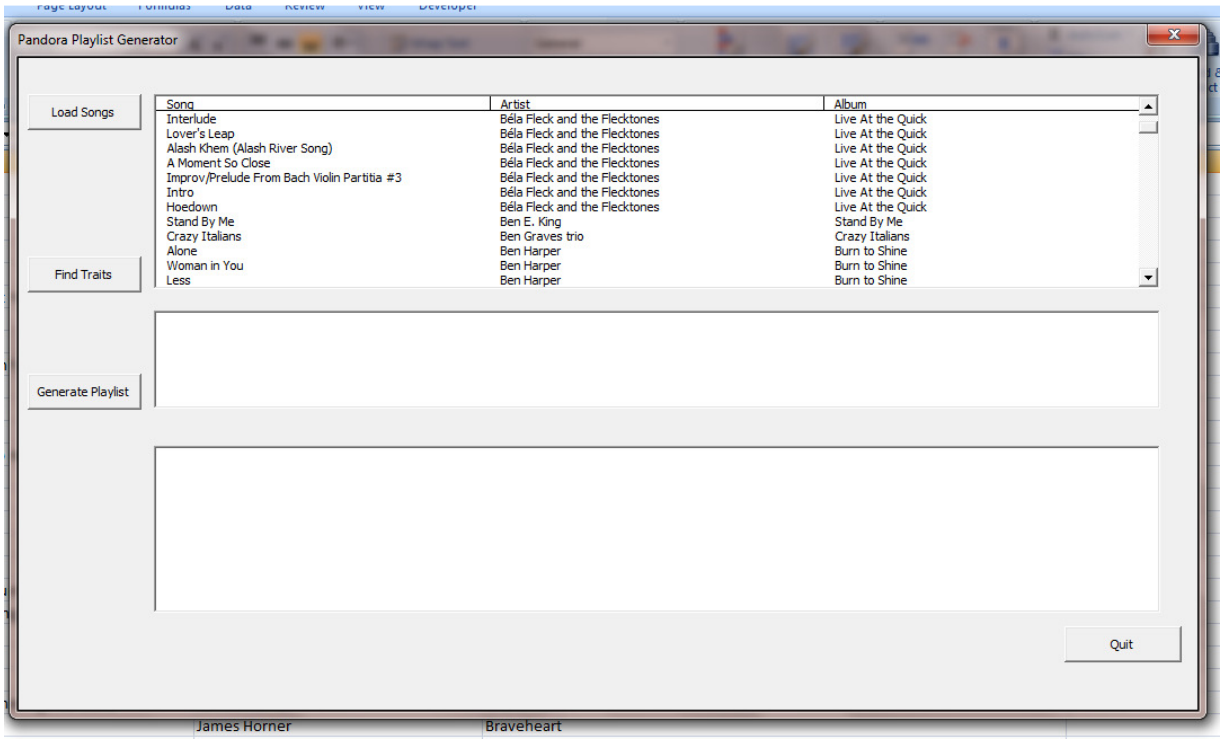
The first step in my program was to allow the user to load songs from his or her iTunes library. Producing a file was simple enough; the 'File' section of the iTunes menu allows the user to export the library to an XML file. The actual XML file itself was (not surprisingly) a rather dense collection of information, only some of which I actually wanted. A brief portion of the file is shown below:



Fortunately, we had learned how to read files line by line, so I simply created a loop in my program that read lines until it found a line with "<key>Name," then used the mid function to find the portion of that line with the song name. I used the same process to find the album and

artist information. I found a specific set of lines that corresponded with the end of the song list in the XML file and set the do/while loop to exit when it hit that spot. As the loop continues, the program saves each combination of artist, album, and song to a sheet in the workbook.

Now that I had this algorithm in place, I created a button which allowed the user to select a specific folder; the program would then find all XML files (that is, files ending with "XML") and run the algorithm described above. Once the songs were loaded in the workbook, I also created procedures that sorted the songs (basically drawn from me recording a macro) by artist and then album name, as well as a procedure that removed any duplicated songs.[1] Finally, I used the rowsource[2] listbox property to load the songs in the workbook into a listbox on the user form that represented the core of my program.



Finding Song Traits

With the songs read and loaded into the program, the hardest part by far was finding song traits for each song. I began by doing some exploring on the Pandora website, and found that each artist had its own page, such as http://www.pandora.com/music/artist/jack+johnson. The page looks as follows:

---

[1] At first, I had the program check for duplicates after reading each song. I soon learned that doing so created an exponentially long lag time; by the time the song number reached the hundreds, each new song meant an examination of the hundreds of songs already loaded.

[2] The beauty of the rowsource property is that it allows me to put column headers in the listbox (song, artist, album).

Notably, as can be seen in the screenshot above, the artist page lists a discography of albums by that artist. Clicking on a specific album leads to the following webpage:

Finally, and most importantly, clicking on a specific song leads to an individual page for the chosen song, which includes a selection of Pandora's song traits:

These traits are the core of what my program needs.

So, the task now was to make my program take a song, with accompanying information for artist and album, and find the specific song page above. Since the trail begins by identifying the right artist page, I tried to take a string with the artist name and create the right link for the artist page. As the page had this form -- http://www.pandora.com/music/artist/jack+johnson -- I began my replacing any spaces with the "+" character. Eventually, I added other modifications in hopes of finding the right page:

- If the artist name had ", The" at the end, I moved the "The" to the beginning.
- I removed anything after a "&" character; thus, Ben Harper & The Innocent Criminals became just Ben Harper.
- I replaced the "é" character (such as found in Michael Bublé or Beyoncé) with the simple "e" character.
- I removed the word "the" from artist names.
- I removed periods from artist names.

With these modifications done, I used the agent class to open my "best guess" at the artist page link. I made sure to handle the case where a file not found or server error occurred, in which case my program would simply report that song traits could not be found. Once I had the

artist page loaded, I used the agent's moveto and gettext functions to identify the section of source code that had the desired album and album link. A short excerpt of the page code is below:

```
<DIV id=album_grid_start>
<DIV style="CLEAR: both"><SPAN class=list_album><A
href="http://www.pandora.com/music/album/barenaked+ladies/all+in+good+time"><
IMG border=0 src="http://cont-sjl-
1.pandora.com/images/public/amz/1/2/7/1/5099962891721_500W_493H.jpg" width=80
height=80></A><BR><A
href="http://www.pandora.com/music/album/barenaked+ladies/all+in+good+time">A
ll In Good Time</A><BR>2010 </SPAN><SPAN class=list_album><A
href="http://www.pandora.com/music/album/barenaked+ladies/barenaked+ladies+ar
e+men"><IMG border=0 src="http://cont-sjl-
1.pandora.com/images/public/amz/0/2/7/4/093624324720_500W_444H.jpg" width=80
height=80></A><BR><A
<DIV id=community_section>
<DIV id=community_section_title>Similar Artists </DIV><A
href="http://www.pandora.com/music/artist/blues+traveler">Blues
Traveler</A><BR><A
href="http://www.pandora.com/music/artist/counting+crows">Counting
Crows</A><BR><A
href="http://www.pandora.com/music/artist/sister+hazel">Sister
Hazel</A><BR><A
href="http://www.pandora.com/music/artist/weezer">Weezer</A><BR><A
href="http://www.pandora.com/music/artist/matchbox+20">Matchbox
Twenty</A><BR></DIV></DIV>
<SCRIPT type=text/javascript>
```

As can be seen from the code above, I used the moveto function to move to album_grid_start, then read links and the corresponding album titles until I got a match or until I hit a link that did not have the right artist (that is, one of the "Similar Artists" displayed after the discography).

Once I had the right link, I had the agent follow that link to the album page. Again, the next step was a matter of examining the source code, finding the portion of the page that listed the songs in the album (found by using the moveto function to find the text "Song Details"), and then reading each song and accompanying link until I found the desired song. If the moveto function returned false, then I knew that the song was not found on the page.

Finally, after directing the agent to open the right song page (if found), I again looked through the source code to find the section of the page with Pandora's song traits (found by using the moveto function and the text "Features of This Song" and then read each trait until no more could be found. If the moveto function returned false, then I knew that the Pandora song page lacked song traits.

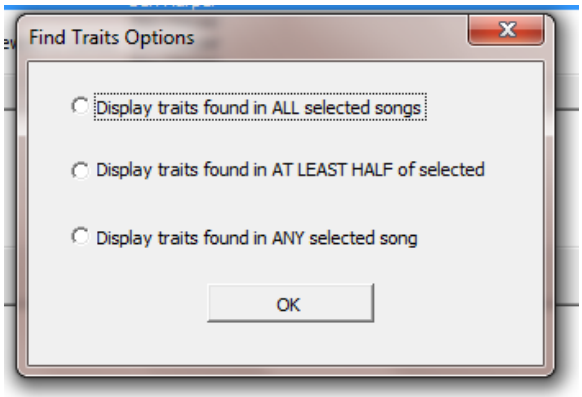
To help the user have an idea of how many traits were found, I recorded the song/artist/album information for all songs for which traits were not found. I then created an additional form, consisting primarily of a single listbox, which was populated (via the rowsource property) with the songs without traits. This form popped up whenever the program was done searching for traits, whether for songs selected by the user or the whole library.

Unfortunately, the actual process of finding traits for each song online takes a long time for Excel. In hopes of speeding up the process, I saved the results of each song's search for traits in the workbook as an archive. That way, before looking online, the program can read through the archive first—if the right song/artist/album is found, then no online searching is necessary. I similarly used a string variable to remember those artists for which the artist page could not be found; if a certain song's artist was found in this string, then the program similarly wouldn't bother looking online for traits.

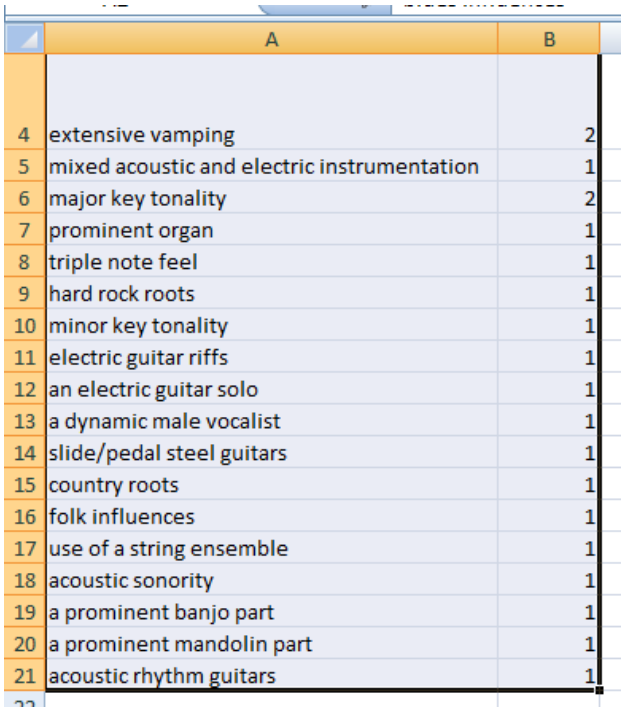
Overall, this part of the project was definitely the most difficult. At first, the agent class I was using had a bug in it that resulted in the agent variable reading the wrong part of a given page's source code. Once that was fixed, it was still difficult to make the best possible "best guess" at an artist page. Still, after much trial and error, I was able to find song traits (if available) for the vast majority of songs in my iTunes library.

Creating the Playlist

Once I could find traits for the loaded songs, the rest of the program consisted of using this algorithm to find traits for songs selected by the user and then search all loaded songs to find a match. I enabled the multiselect feature of the loaded songs list box so that the user could select multiple songs to create a list of potential traits for the playlist. Once the user selected the "Find Traits" button, the program used the process described above to find the traits for each selected song and then count the number of times each trait appeared. First, however, I created an additional form that asked the user if it should display traits found in all of the selected songs, at least of the selected songs, or any of the selected songs, as shown below:

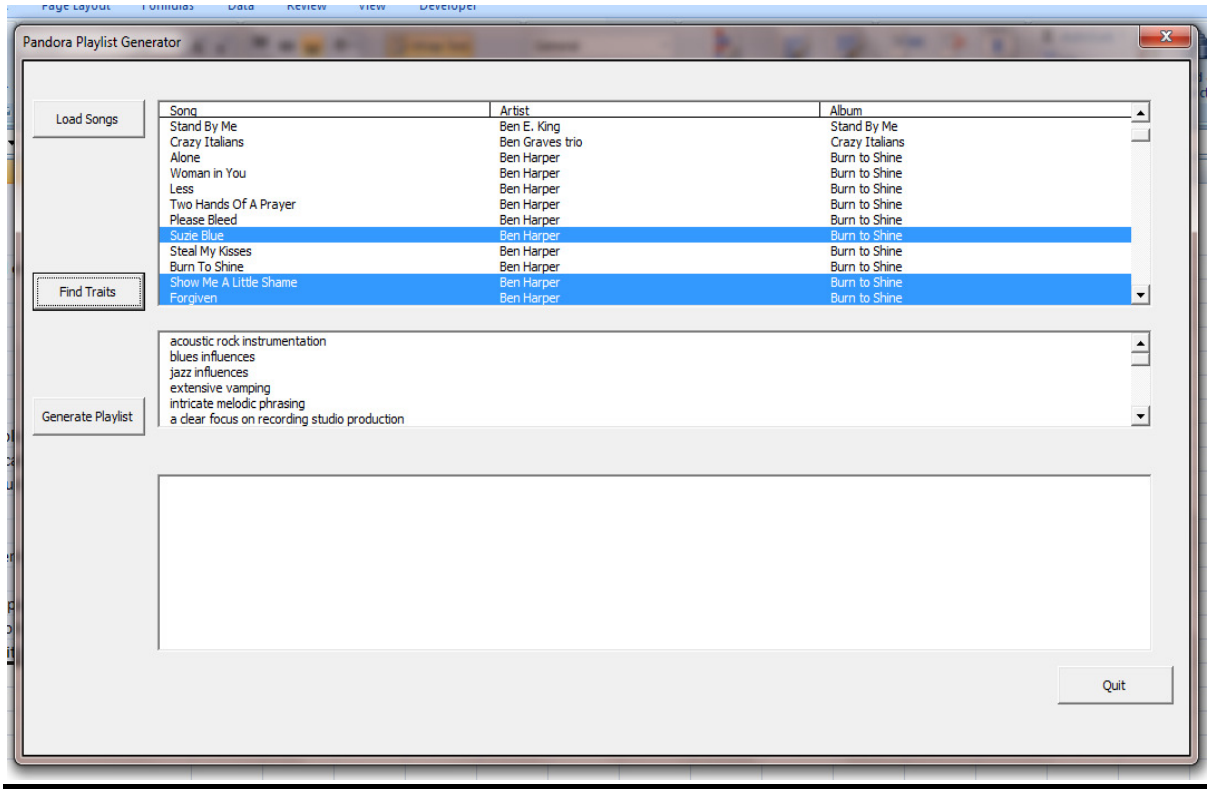


The result of the counting of traits was saved in the workbook, as seen below:



	A	B
4	extensive vamping	2
5	mixed acoustic and electric instrumentation	1
6	major key tonality	2
7	prominent organ	1
8	triple note feel	1
9	hard rock roots	1
10	minor key tonality	1
11	electric guitar riffs	1
12	an electric guitar solo	1
13	a dynamic male vocalist	1
14	slide/pedal steel guitars	1
15	country roots	1
16	folk influences	1
17	use of a string ensemble	1
18	acoustic sonority	1
19	a prominent banjo part	1
20	a prominent mandolin part	1
21	acoustic rhythm guitars	1

Based on the user's choice in the form described above, the program then determined which traits met the cutoff for display. Any traits that met this cutoff were added to a listbox in the main form, as seen below:



The last step of the program was to search the entire library for songs with traits that included the traits selected by the user. I again enabled the multiselect property of the traits listbox and saved the selected traits to an array. Once the user clicked the "Generate Playlist" button, I made the program go through every song in the top listbox, finding the traits for each song and checking to see if the selected traits were included.[3] If a given song did have the selected traits, then the song was added to the bottom listbox, as shown below:

Opening/Closing the Program

Because I didn't want the user to be able to access the actual sheets in the workbook (used for creating the listboxes, saving the archives, etc.), I added code to the open workbook event that loaded the main user form. Similarly, when the "Quit" button is clicked, I made the workbook save and then close.

[3] Specifically, when traits were found, they were recorded as a single string with each trait set off by a "~" character. That way, the program merely used the instr function to see if a particular trait was in the set of traits for each song.

LEARNING/CONCEPTUAL DIFFICULTIES

Lessons Learned

In particular, I learned through this project how to navigate and access information on the web through VBA. The approach I used—first finding a specific artist page, then the right album page, then the right song page, then finding the required traits—meant that I had to wade through several different pages with accompanying source code. I learned how to familiarize myself with the agent class (especially the "moveto" and "gettext" functions), to the point that I felt very comfortable looking through source code and identifying (and gathering) the exact information that I needed.

I also learned certain techniques for speeding up my program. Unfortunately, finding song traits online for hundreds (or even thousands) of songs is a necessarily slow process. However, I found alternative methods—like saving songs with their accompanying traits to an archive in the workbook—that helped alleviate the slow search time. I also learned that seemingly innocuous searches—like looking for duplicates in the loaded song list as each new song was loaded from the XML file—could mean an exponential growth in lag time. I wish my program could be faster when first finding song traits online, but I feel that at least I took some significant steps in speeding things up where I could.

Conceptual Difficulties

By far, the most difficult part of my project was finding song traits for songs that had unusual artist names. My program was designed to make a "best guess" at the actual artist page on Pandora's site (e.g., http://www.pandora.com/music/artist/jack+johnson); unfortunately, it was difficult to come up with "all purpose" steps that would result in the right page. At this point, my program still doesn't successfully find an artist page for every single artist with information on Pandora. I could probably make steps to "fix" artist names indefinitely (for example, telling my program that country band Big & Rich would be found at http://www.pandora.com/music/artist/big+rich+country), but something would always be slipping through the cracks. Nonetheless, I feel satisfied with the number of artists that the program does successfully identify; specifically, out of 1,671 total songs in my music library, the program found song pages for 883 songs.

Other elements did not make it to my program because of simple time constraints. I wanted my program to actually create a playlist file that could be imported into iTunes, but based on what I expected the task would take, I felt that I didn't have time to make it work satisfactorily. I also wanted to allow the user to load in bookmarked songs from his or her own personal Profile page, but as things went down to the wire I felt that time would be better spent on improving my program's ability to find song traits for songs in the user's iTunes library.

One other note regarding conceptual difficulties: unfortunately, I spent a good deal of time trying to navigate the Pandora webpages without knowing that my class agent was fatally flawed by a bug. I was using an earlier version of the agent that would, in a nutshell, often examine the wrong part of the page's source code. Much time was spent trying to hit on the right information in accessed webpages (made impossible by the bug) and trying to figure out alternative ways of getting the required information. Once the bug was fixed, I was able to proceed with finding the traits, etc. from the accessed webpages, but only after a great deal of time was lost.