Generating iTunes Related-Artist Playlists

Spencer Robinson | ISYS 540 | Winter 2010

Executive Summary

Music fans and critics often speak about musical artists in terms of which other artists are similar. When reading about an unfamiliar artist, knowing which artists are similar to the new artist can help a listener to identify attributes the unfamiliar artist will have.

I have created an Excel-based program that will allow a user to enter a musical artist, either in the user's iTunes library or not, and present the user with a list of similar artists. Additionally, the program searches the user's iTunes music library for songs by the artists in the list and generate a playlist with all the songs in the user's library recorded by artists similar to the artist the user searched.

I got the idea for this project from a similar project created by Shawn Hansen for the ISYS 540 class in Fall Semester 2009. I used one "This is how to export an iTunes playlist" instructional picture from his project, but all of the code included in this project is my own; I didn't use any code from Shawn's project in completing my own.

Implementation Documentation

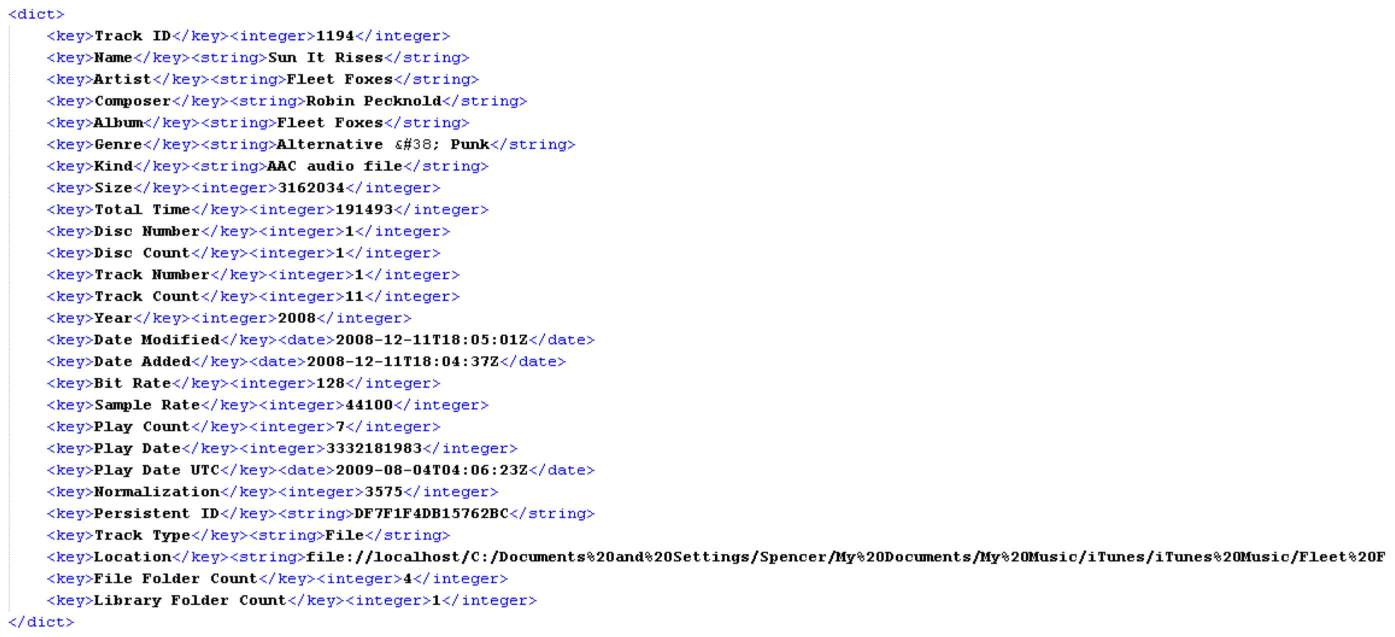
To use the program, the user first opens the "Music Project.xlsm" spreadsheet. The user enters the name of the artist he wants to search for in cell A1 of the "Artists" worksheet, and clicks the "Click to Find" button. The program then generates the playlist file with all songs in the library by the artist and the related artists.

Completing the objectives of the project requires five tasks:

1. Reading the user's iTunes library from an XML file.
2. Generating a database from the user's library.
3. Retrieving a list of similar artists based on the user's input of an artist to search for.
4. Searching the database for songs recorded by the artists in the list.
5. Creating a file with the songs found that iTunes can import as a playlist.

Reading the User's iTunes Library from XML file

Before using the program, the user must export his iTunes library (this must be done using the iTunes interface). The library is exported as an XML file that, generally, looks like this (of course, this structure is repeated for every song in the library, about 1,000 songs on my computer):



```
<dict>
    <key>Track ID</key><integer>1194</integer>
    <key>Name</key><string>Sun It Rises</string>
    <key>Artist</key><string>Fleet Foxes</string>
    <key>Composer</key><string>Robin Pecknold</string>
    <key>Album</key><string>Fleet Foxes</string>
    <key>Genre</key><string>Alternative &#38; Punk</string>
    <key>Kind</key><string>AAC audio file</string>
    <key>Size</key><integer>3162034</integer>
    <key>Total Time</key><integer>191493</integer>
    <key>Disc Number</key><integer>1</integer>
    <key>Disc Count</key><integer>1</integer>
    <key>Track Number</key><integer>1</integer>
    <key>Track Count</key><integer>11</integer>
    <key>Year</key><integer>2008</integer>
    <key>Date Modified</key><date>2008-12-11T18:05:01Z</date>
    <key>Date Added</key><date>2008-12-11T18:04:37Z</date>
    <key>Bit Rate</key><integer>128</integer>
    <key>Sample Rate</key><integer>44100</integer>
    <key>Play Count</key><integer>7</integer>
    <key>Play Date</key><integer>3332181983</integer>
    <key>Play Date UTC</key><date>2009-08-04T04:06:23Z</date>
    <key>Normalization</key><integer>3575</integer>
    <key>Persistent ID</key><string>DF7F1F4DB15762BC</string>
    <key>Track Type</key><string>File</string>
    <key>Location</key><string>file://localhost/C:/Documents%20and%20Settings/Spencer/My%20Documents/My%20Music/iTunes/iTunes%20Music/Fleet%20F
    <key>File Folder Count</key><integer>4</integer>
    <key>Library Folder Count</key><integer>1</integer>
</dict>
```

I chose to use the standard VBA file-reading method (e.g., `Open sFileName For Input As iFileNum`). When the program finds the "Track ID" line, it starts reading in the information for the song. It then searches for the lines that indicate the title, artist, album, and file location for the song, and puts those into the database for each song (described in the next section). When the program reaches the end of the information for each song, it again begins searching for the start of the next song's information. Once it reaches the end of the information for the library's songs, the program closes the file.

Generating the Database

Before the library XML file is read into the program, the database for the songs is created. The database file "Music.mdb" is created beforehand, and is included with the installation of the program. The program uses an ADODB Connection and recordset to deal with the database:

```
Set dbConn = CreateObject("ADODB.Connection")
    dbConn.Open ("Driver={Microsoft Access Driver (*.mdb)};" & _
      "DBQ=Music.mdb;" & _
      "DefaultDir=C:\;" & _
      "Uid=Admin;Pwd=;")
```

First, the program drops the "Songs" table to get rid of any outdated data. Then, the program creates a new table for songs:

```
sql = "CREATE TABLE [Songs] (TrackID INTEGER PRIMARY KEY, Title VARCHAR(100), Artist
VARCHAR(100), Album VARCHAR(100), Location VARCHAR(200));"
dbConn.Execute (sql)
```

Then, while parsing the library file, each time the program reaches the end of the information for a song, it creates a record in the table for that song using the information just parsed:

```
sql = "INSERT INTO [Songs] (TrackID, Artist, Album, Title, Location) " & _
                   "VALUES (" & TrackID & ", '" & Artist & "', '" & Album & "', '" &
                   Title & "', '" & Location & "')"
 dbConn.Execute (sql)
```

Retrieving a list of similar artists from the web

While there are many, many services that will generate a list of similar artists based on an artist name input by the user, I decided to use http://www.tastekid.com/ for my project because of ease of automating the search function and good, concise lists of similar artists.

To retrieve the information needed from the website, the program first generates an instance of Internet Explorer (`Set ie = CreateObject("internetexplorer.application")`). The program then sends this instance to http://www.tastekid.com/ask?f=1&q= with the artist's name appended. The Internet Explorer instance retrieves all the information from the page returned and closes.

The program then parses this HTML document similar to the way it parsed the library XML file earlier. However, this time the program puts the names of the artists in cells in the spreadsheet instead of creating a database for them.

Searching the database for artists in the list

To find songs by artists in the list returned from the web, the program uses the ADODB connection:

```
sql = "SELECT * FROM [Songs] WHERE ARTIST='" & UCase(Artist) & "';"
Set rs = dbConn.Execute(sql)
```

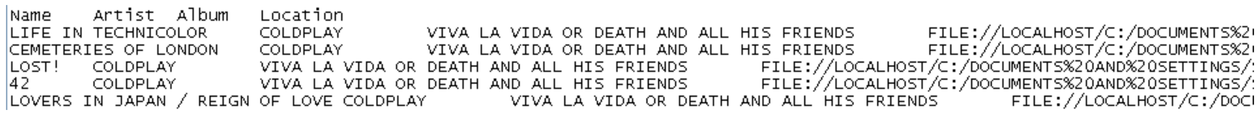
The program repeats this action for every artist in the list, placing the information for title, artist, album, and file location in a new worksheet. Also, the program places an indication next to the artist names in the original worksheet telling whether any songs by each artist are present in the library.

Creating a playlist file with the appropriate songs

Now that the program has all of the songs from the artists included in the list on a worksheet, it can create a file iTunes can import as a playlist.

One of the specifications iTunes can use to import as a playlist is a tab-delimited text file. This file can contain many pieces of information for each song included, but only the title, artist, album, and file location are needed. The program uses the VBA standard output file method (`Open fileName For Output As #iFileNum`) where the file name is the original artist's name followed by the phrase "related artists". The file is a simple .txt file.

The program writes a row of headers to the file, and then for each song included on the playlist worksheet, it writes a line with the songs included, separated by tabs. The final output of the file looks like this:



```
Name    Artist  Album   Location
LIFE IN TECHNICOLOR     COLDPLAY        VIVA LA VIDA OR DEATH AND ALL HIS FRIENDS       FILE://LOCALHOST/C:/DOCUMENTS%2
CEMETERIES OF LONDON    COLDPLAY        VIVA LA VIDA OR DEATH AND ALL HIS FRIENDS       FILE://LOCALHOST/C:/DOCUMENTS%2
LOST!   COLDPLAY        VIVA LA VIDA OR DEATH AND ALL HIS FRIENDS       FILE://LOCALHOST/C:/DOCUMENTS%20AND%20SETTINGS/
42      COLDPLAY        VIVA LA VIDA OR DEATH AND ALL HIS FRIENDS       FILE://LOCALHOST/C:/DOCUMENTS%20AND%20SETTINGS/
LOVERS IN JAPAN / REIGN OF LOVE COLDPLAY        VIVA LA VIDA OR DEATH AND ALL HIS FRIENDS       FILE://LOCALHOST/C:/DOC
```

While this format isn't the easiest on the eyes, iTunes has no problem reading this file. After the file is created, the user can open iTunes and import the playlist from the file. The file is given the file name "[Artist name] Related Artists.txt" and is saved to the root of the C:/. The user can also continue to search for artists to find similar artists in his iTunes library.

Lessons Learned, Difficulties Dealt With

The most important lesson I learned (and the biggest difficulty I dealt with in completing the project) was the importance of being tidy in dealing with inputs and outputs. The program both writes to and reads from text files, and during the course of creating the program I encountered numerous "file already open" errors. I struggled with these errors for a great deal of time before determining they were occurring because I simply wasn't closing files properly after reading from or writing to them. I encountered similar problems with operating Internet Explorer from code—I didn't close the instances of Internet Explorer, causing several problems. It slowed down my performance and created problems for my Internet Explorer cache. Once I realized what the problem was, I searched down the `ie.Quit` command and that solved the problem.

Another important lesson I learned was using escape characters while working with SQL. Of course, SQL uses the single-quote character, so saving a band's name that has an apostrophe (usually input as a single-quote) caused problems. So I researched the issue and found that two single quotes escapes as a single-quote in SQL, solving the problem. A similar problem occurred with the "&" character—the iTunes library escapes the & as &38; while the website I got information from escapes the & character as

&amp;. Simply replacing these escape characters with the associated “real” character cleared up problems with escaping.

A final problem I encountered while completing the project was with creating a playlist file iTunes could read in. iTunes can import .txt files and .xml files as playlists, and originally I tried to use the XML format, since I’m quite familiar with it. However, this format includes information I could not determine how to obtain, such as a playlist persistent ID (which I would have had to create). So I went back and saw that the text-file method was much easier, so I went with that. I wasn’t sure how to format the file, but the example I exported from iTunes looked tab-delimited, so I tried that, and it worked. The lesson I learned from this experience is, it’s generally better to go the simpler route to solve a problem, all else being equal.