Find Orphan Song Files in iTunes

In the last blog post you learned to read songs from an exported iTunes XML file. If you have been using iTunes for a long time and have deleted songs, merged songs from other libraries, moved your library from one computer to another, then you may not know it, but there could song files on your hard drive that are no longer in the iTunes library. This blog post shows you how to locate those missing files. To follow along with this blog post, read and follow the instructions in the first blog post on reading songs from iTunes.

Overview of Orphan Finding

An orphan file is a file that is located within your music folder but is not in the iTunes library. To locate these files, you need to do the following:

- Export the current iTunes library as an XML file
- Read all songs from the XML file
- Retrieve the music folder location from the XML file
- Read all files located in the music folder (and all subdirectories)
- Compare files on disk with those in the XML file
- Determine if the file is an audio/video file
- Add files missing from XML file to a collection
- Display collection of missing files

Apple Orphan File Class

To track which files are on disk, but not in the iTunes library, create a new class to capture the path and file name. Create a property to also track whether this file is a valid audio/video file, or another type of file. There might be other types of files under the music folder because iTunes put them there, or maybe you accidentally copied some other files into that folder. Add the following class to the Apple iTunes class library project.

```
public class OrphanFile
{
  public bool IsAudioVideoFile { get; set; }
  public string Location { get; set; }
}
```

Apple Song Orphan Finder Class

In the first blog post, you created a class named AppleSongReader to read all files in the iTunes XML file. Instead of adding functionality to this class to locate orphan files, add a new class named AppleSongOrphanFinder to the Apple iTunes class library project. Set this class to inherit from the AppleSongReader class. You can add the functionality to find orphan files to this new class. In Figure 1 you see the new AppleOrphanFinder class and the OrphanFile class that have been added to the Apple iTunes class library project.

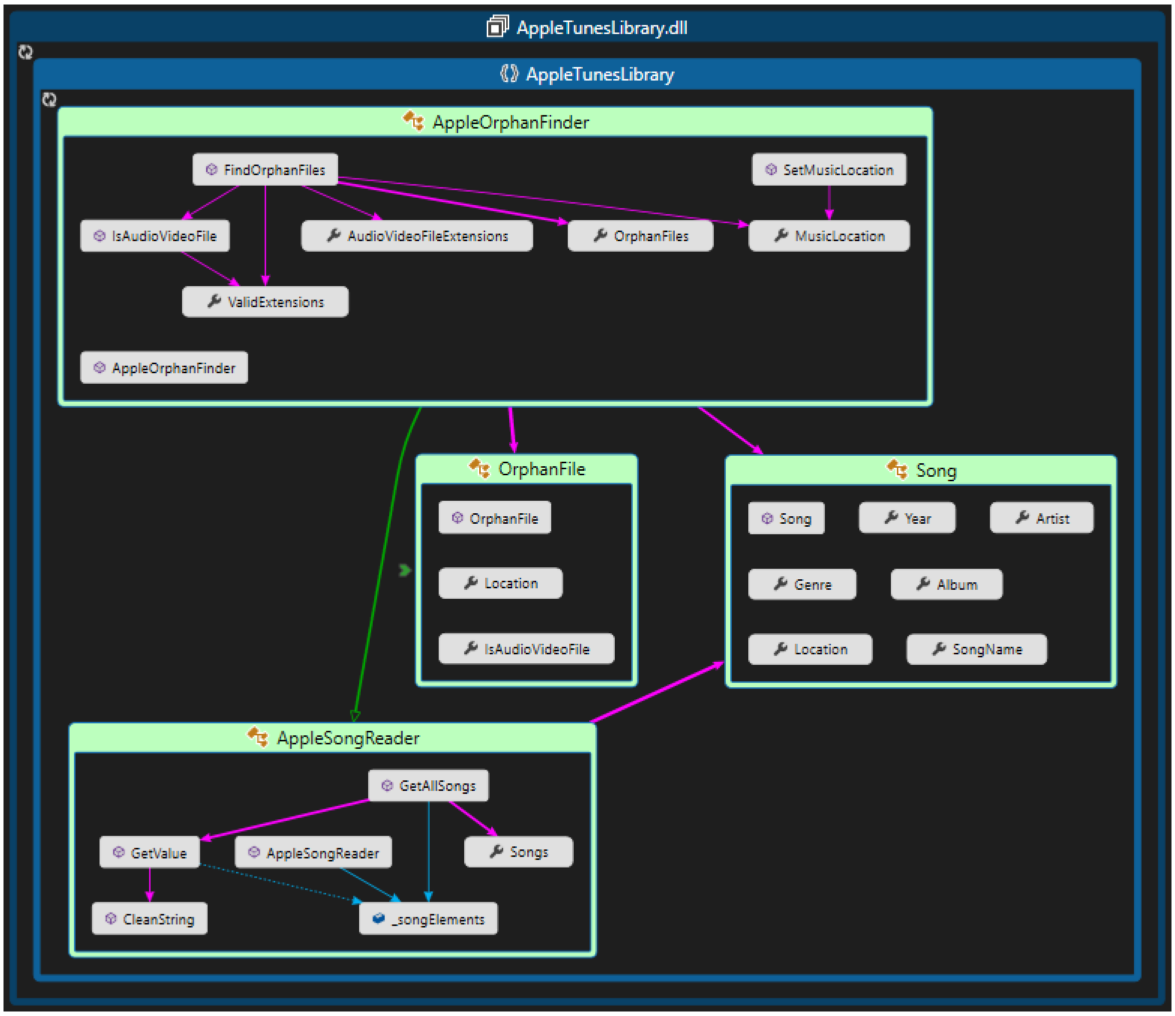


Figure 1: Add two new classes to track orphan songs to the Apple iTunes Library DLL

The following table describes the properties and methods in the AppleOrphanFinder class.

Properties	**Description**
OrphanFiles	The collection of orphan files found.
MusicLocation	The top-level folder where all music files are located.
AudioVideoFileExtensions	A string with a comma-delimited list of valid audio and video file extensions. Valid extensions are those such as .m4p, .m4a, .m4v, .mp3, .mp4, and .wav. You may add other extensions as needed.
ValidExtensions	A collection of audio and video file extensions. This property is created from the AudioVideoFileExtensions property.

Methods	Description
SetMusicLocation	Call this method to read the Library.xml file and find the top-level folder where all your music is stored. You must call this method prior to calling the FindOrphanFiles() method.
FindOrphanFiles	Call this method after loading all songs from the Library.xml file to locate all orphan files in the MusicLocation folder.
IsAudioVideoFile	This method is used to set the IsAudioVideoFile property on each OrphanFile object.

Structure of the AppleOrphanFinder Class

Now that you know the structure of this class, let's create the AppleOrphanFinder class. Add the properties to the AppleOrphanFinder class. Create a stub of each method as well.

```
public class AppleOrphanFinder : AppleSongReader
{
  public List<OrphanFile> OrphanFiles { get; set; }
  public string MusicLocation { get; set; }
  public string AudioVideoFileExtensions { get; set; }
  public List<string> ValidExtensions { get; set; }

  public virtual void SetMusicLocation(string libraryFile)
  {
  }

  public List<OrphanFile> FindOrphanFiles()
  {
    return OrphanFiles;
  }

  protected virtual bool IsAudioVideoFile(string file)
  {
    bool ret = true;

    return ret;
  }
}
```

Set Music Location Method

In iTunes you can set the top-level folder where all your music files are located. Apple stores this folder name within the first <dict> element in the XML file as shown in the code below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
 <key>Major Version</key><integer>1</integer>
 <key>Minor Version</key><integer>1</integer>
 <key>Date</key><date>2018-08-08T19:38:14Z</date>
 <key>Application Version</key><string>12.8.0.150</string>
 <key>Features</key><integer>5</integer>
 <key>Show Content Ratings</key><true/>
 <key>Music Folder</key><string>file://localhost/P:/Music/</string>
 <key>Library Persistent ID</key><string>0472F8AA57FD57AE</string>
 <key>Tracks</key>
    // MUSIC TRACKS ARE BELOW HERE
</dict>
```

Write the code in the SetMusicLocation() method to read the XML library into an XElement object. Locate the first <dict> element and then read all the <key> elements within that element. Convert all <key> elements into a list and store into the *_songElements* property. You can now use the GetValue() method to retrieve the Music Folder key and get the value of the top-level music folder.

```
public virtual void SetMusicLocation(string libraryFile)
{
  // Load iTunes XML library
  XElement doc = XElement.Load(libraryFile);

  // Get music folder key
  IEnumerable<XElement> node =
          from dict in doc.Elements("dict")
                          .Elements("key")
          select dict;

  // Set song elements so you can use the GetValue() method
  _songElements = node.ToList();

  // Set the music location property
  MusicLocation = GetValue<string>("Music Folder", "Unknown");
}
```

Is Valid Audio/Video File Method

The OrphanFile class has a property named *IsAudioVideoFile*. To set this property, check the file extension on each file read from the disk to see if it is contained within a set of valid audio/video file extensions. The list of valid extensions must be set prior to calling the FindOrphanFiles() method. While this method is essentially one line of code, I broke it out into a separate method in case you wish to do some other kind of file checking to determine if it is a valid file.

```
protected virtual bool IsAudioVideoFile(string file)
{
  bool ret = true;

  ret = ValidExtensions.Contains(Path.GetExtension(file));

  return ret;
}
```

Find Orphan Files Method

Prior to calling the FindOrphanFiles() method make sure you call the SetMusicLocation() method, and set the *AudioVideoFileExtensions* property. The FindOrphanFiles() method reads all files from the *MusicLocation* property and all subdirectories under this folder. It then loops through each file found and checks if that file is in the *Songs* collection. If the file is not found, a new OrphanFile object is created. The *IsAudioVideoFile* property is set from the results of calling the IsAudioVideoFile() method. The *Location* property is set to the actual path and file name. This new OrphanFile object is added to the *OrphanFiles* property. After all files have been processed, the *OrphanFiles* collection is sorted in descending order on the *IsAudioVideoFile* property so all audio/video files are displayed first.

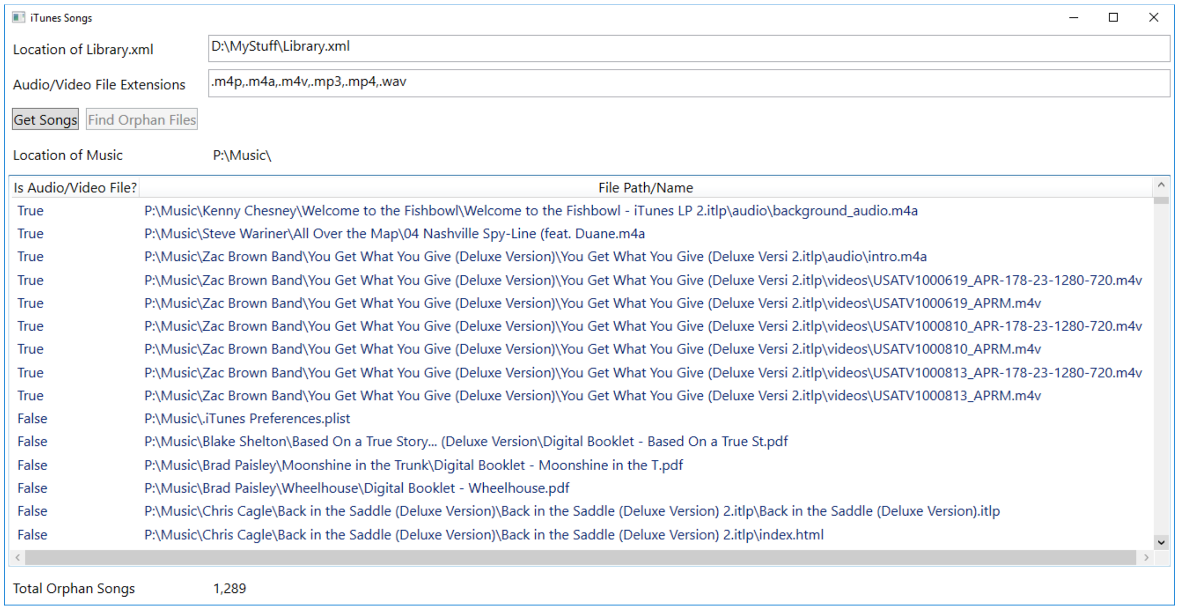
```
public List<OrphanFile> FindOrphanFiles()
{
  OrphanFile orphan;
  string[] files;
  string fileLower;

  // Get all files within music folder
  files = Directory.GetFiles(MusicLocation, "*.*",
SearchOption.AllDirectories);

  // Clear OrphanFiles collection
  OrphanFiles = new List<OrphanFile>();
  // Create array of valid audio/video extensions
  ValidExtensions = AudioVideoFileExtensions.Split(',').ToList();

  // Loop through all files
  foreach (string file in files) {
    // Convert file to lower case
    fileLower = file.ToLower();
    // Locate file within Songs collection
    var tmp = Songs.Find(s => s.Location.ToLower() == fileLower);
    // Did we find the song?
    if (tmp == null) {
      // Create a new OrphanFile object
      orphan = new OrphanFile();
      orphan.IsAudioVideoFile = IsAudioVideoFile(file);
      orphan.Location = file;
      // Add orphan file to collection
      OrphanFiles.Add(orphan);
    }
  }

  // Sort songs by artist
  OrphanFiles = OrphanFiles.OrderByDescending(s =>
s.IsAudioVideoFile).ToList();

  return OrphanFiles;
}
```

Modify WPF Application

You are now ready to modify the WPF application to display all orphan files. After running this process, you may see a list of orphan files as shown in Figure 2. The changes made to this window compared to the WPF window in the first blog post are as follows:

- A new row is added to accept a comma-delimited list of audio/video file extensions.
- A new button "Find Orphan Files" is added.
- A new row is added to display the folder of where the music is located.

- A new ListView control is added to display the *OrphanFiles* collection.



iTunes Songs

Location of Library.xml D:\MyStuff\Library.xml

Audio/Video File Extensions .m4p,.m4a,.m4v,.mp3,.mp4,.wav

Get Songs | Find Orphan Files

Location of Music P:\Music\

Is Audio/Video File?	File Path/Name
True	P:\Music\Kenny Chesney\Welcome to the Fishbowl\Welcome to the Fishbowl - iTunes LP 2.itlp\audio\background_audio.m4a
True	P:\Music\Steve Wariner\All Over the Map\04 Nashville Spy-Line (feat. Duane.m4a
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\audio\intro.m4a
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000619_APR-178-23-1280-720.m4v
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000619_APRM.m4v
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000810_APR-178-23-1280-720.m4v
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000810_APRM.m4v
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000813_APR-178-23-1280-720.m4v
True	P:\Music\Zac Brown Band\You Get What You Give (Deluxe Version)\You Get What You Give (Deluxe Versi 2.itlp\videos\USATV1000813_APRM.m4v
False	P:\Music\.iTunes Preferences.plist
False	P:\Music\Blake Shelton\Based On a True Story... (Deluxe Version\Digital Booklet - Based On a True St.pdf
False	P:\Music\Brad Paisley\Moonshine in the Trunk\Digital Booklet - Moonshine in the T.pdf
False	P:\Music\Brad Paisley\Wheelhouse\Digital Booklet - Wheelhouse.pdf
False	P:\Music\Chris Cagle\Back in the Saddle (Deluxe Version)\Back in the Saddle (Deluxe Version) 2.itlp\Back in the Saddle (Deluxe Version).itlp
False	P:\Music\Chris Cagle\Back in the Saddle (Deluxe Version)\Back in the Saddle (Deluxe Version) 2.itlp\index.html

Total Orphan Songs 1,289

Figure 2: Display a list of orphan files

XAML for WPF Window

Below is the complete XAML for building the WPF window shown in Figure 2. The changes that were made are outlined in bold text.

```
<Window x:Class="iTunesOrphanFinder.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:iTunesOrphanFinder"
        mc:Ignorable="d"
        FontSize="16"
        Title="iTunes Songs"
        WindowStartupLocation="CenterScreen">
  <Window.Resources>
    <Style TargetType="Label">
      <Setter Property="Margin"
              Value="4" />
    </Style>
    <Style TargetType="TextBox">
      <Setter Property="Margin"
              Value="4" />
    </Style>
    <Style TargetType="StackPanel">
      <Setter Property="Margin"
              Value="4" />
    </Style>
    <Style TargetType="Button">
      <Setter Property="Margin"
              Value="4" />
    </Style>
    <Style TargetType="ListView">
      <Setter Property="Margin"
              Value="4" />
    </Style>
  </Window.Resources>
  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="Auto" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Label Grid.Column="0">Location of Library.xml</Label>
    <TextBox x:Name="txtFileName"
             Grid.Column="1"></TextBox>
    <Label Grid.Row="1">Audio/Video File Extensions</Label>
    <TextBox x:Name="txtFileExtensions"
             Grid.Row="1"
             Grid.Column="1"></TextBox>
    <StackPanel Grid.Row="2"
                Orientation="Horizontal">
      <Button x:Name="btnGetSongs"
              Content="Get Songs"
              Click="btnGetSongs_Click" />
      <Button x:Name="btnFindOrphans"
              IsEnabled="False"
              Content="Find Orphan Files"
              Click="btnFindOrphans_Click" />
    </StackPanel>
    <Label Grid.Row="3">Location of Music</Label>
    <Label x:Name="lblMusicLocation"
           Grid.Row="3"
           Grid.Column="1" />
    <ListView x:Name="lstSongs"
              ItemsSource="{Binding}"
              ScrollViewer.HorizontalScrollBarVisibility="Visible"
              ScrollViewer.VerticalScrollBarVisibility="Visible"
              Grid.Row="4"
              Grid.ColumnSpan="2">
      <ListView.View>
        <GridView>
          <GridViewColumn Header="Song Name"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=SongName}" />
          <GridViewColumn Header="Artist"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=Artist}" />
          <GridViewColumn Header="Album"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=Album}" />
          <GridViewColumn Header="Genre"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=Genre}" />
          <GridViewColumn Header="Year"
                          Width="Auto"
                          DisplayMemberBinding="{Binding Path=Year}"
/>
          <GridViewColumn Header="File Path/Name"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=Location}" />
        </GridView>
      </ListView.View>
    </ListView>
    <ListView x:Name="lstOrphanFiles"
              Visibility="Hidden"
              ItemsSource="{Binding}"
              ScrollViewer.HorizontalScrollBarVisibility="Visible"
              ScrollViewer.VerticalScrollBarVisibility="Visible"
              Grid.Row="4"
              Grid.ColumnSpan="2">
      <ListView.View>
        <GridView>
          <GridViewColumn Header="Is Audio/Video File?"
```

```
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=IsAudioVideoFile}" />
          <GridViewColumn Header="File Path/Name"
                          Width="Auto"
                          DisplayMemberBinding="{Binding
Path=Location}" />
        </GridView>
      </ListView.View>
    </ListView>
    <Label x:Name="lblSongCount"
           Content="Total Songs"
           Grid.Row="5" />
    <Label x:Name="lblCount"
           Content="0"
           Grid.Column="1"
           Grid.Row="5" />
  </Grid>
</Window>
```

Code Behind for WPF Window

There are a few changes you need to make in the code behind for the WPF window to read all songs and find orphan files. First, add a new field of the type AppleOrphanFinder.

```
AppleOrphanFinder _Reader = new AppleOrphanFinder();
```

Next, modify the code within the btnGetSongs_Click() event procedure to use the _*Reader* field variable instead of an instance of the AppSongReader. Call the SetMusicLocation() method to set the top-level folder for your music. Finally, after all songs are read, enable the "Find Orphan Files" button.

```
private void btnGetSongs_Click(object sender, RoutedEventArgs e)
{
  Mouse.OverrideCursor = Cursors.Wait;
  // Read all iTunes songs
  _Reader.GetAllSongs(txtFileName.Text);
  // Set music folder location
  _Reader.SetMusicLocation(txtFileName.Text);
  Mouse.OverrideCursor = null;

  // Display music location
  lblMusicLocation.Content = _Reader.MusicLocation;
  btnFindOrphans.IsEnabled = true;

  // Place song list into ListView control
  lstSongs.DataContext = _Reader.Songs;

  // Report total songs read
  lblCount.Content = _Reader.Songs.Count.ToString("###,###");
}
```

Locate Orphan Files

After reading all songs from the XML file, the "Find Orphan Files" button becomes enabled. When you click on this button, the code in the btnFindOrphans_Click() event procedure is called. This method checks if the top-level music folder exists. If it does, the *AudioVideoFileExtensions* property is set from the values entered in the "Audio/Video File Extensions" text box. The FindOrphanFiles() method is called and after this method runs, the *lstOrphanFiles* ListView control is made visible and the *lstSongs* ListView control is hidden.

```
private void btnFindOrphans_Click(object sender, RoutedEventArgs e)
{
  if (Directory.Exists(lblMusicLocation.Content.ToString())) {
    btnFindOrphans.IsEnabled = false;
    btnGetSongs.IsEnabled = false;

    Mouse.OverrideCursor = Cursors.Wait;
    // Set valid audio/video file extensions
    _Reader.AudioVideoFileExtensions = txtFileExtensions.Text;

    // Locate orphan songs
    _Reader.FindOrphanFiles();
    Mouse.OverrideCursor = null;

    // Place orphan songs into ListView control
    lstOrphanFiles.DataContext = _Reader.OrphanFiles;
    lstOrphanFiles.Visibility = Visibility.Visible;
    lstSongs.Visibility = Visibility.Hidden;

    // Report orphan songs read
    lblSongCount.Content = "Total Orphan Songs";
    lblCount.Content =
_Reader.OrphanFiles.Count.ToString("###,###");

    btnGetSongs.IsEnabled = true;
  }
  else {
    MessageBox.Show("Directory: " + lblMusicLocation.Content + "
does not exist.");
  }
}
```

Once you have the list of orphan files, you can decide if you want to delete them, or maybe re-add them back into your iTunes library.

Summary

In this blog post you learned to locate orphan files in your iTunes music folder. To locate missing files, you need to read the top-level music folder from the Library.xml file. You then need to read all songs from the XML file and compare this list against all songs you read from the actual files located in the iTunes music folder.

Sample Code

You can download the complete sample code at my website. http://www.pdsa.com/downloads. Choose "PDSA/Fairway Blog", then "Find Orphan Song Files in iTunes" from the drop-down.