

Forum:[Hack Forums](#)

Thread: [\[Tutorial\]The IDE Crasher's Guide To WAV Audio Playback In C# Commandline Environment](#)

KodaKing - May 12, 2018, 11:25 am

The IDE Crasher's Guide To WAV Audio Playback In C# Commandline Environments

Written by -°Koda°-

Within all C# and C++ programs we obviously need to invoke the runtime dependency libraries necessary. Since this is dedicated to running WAV Audio format for Playback in a console or command line terminal, we will need to call System.Console, System.Console, System.Collections.Generic, System.IO, System.Globalization, System.Data, and System.Media as shown following:

```
Code using System.Console;
using System.Collections.Generic;
using System.IO;
using System.Media;
using System.Data;
using System.Globalization;
```

Once this has been completed, move into the body of you active namespace and running class for the audio handling.

To bridge a connection faster, placing directory indexers will allow for a quick jump to your audio section. Mine will be called AudioCat for Audio Category.

Once you have created you index market for your start point, you will need to supply a standard greeting to allow the user to understand any applicable information and begin the dialogue for user input.

```
Code Console.WriteLine("Welcome To Your Section For Audio Functions Within This Command
Line Terminal!");
```

Following the opening line we should get direction for the file path in question. Given that this is only supporting WAV format, and is in use in a command line environment, the path will need to be a perfect line from the C:// folder of your drive to the filename complete with .WAV at the end.

```
Code Console.WriteLine("Please Type The File Path To The Song You Would Like To Play: ");
Here is where the juicy part sits....
```

Now that they are typing in the file path, we need to set up that which is going to direct it to it's media handler event object.

Start by taking the user input in as a strong argument and store it in a variable, for this tutorial mine will be named songPath.

```
Code string songPath = Console.ReadLine();
```

After we have saved the directory, we will need to initialize an instance of the SongPlayer function and parse in our songPath string variable.

When initializing the SongPlayer object we will need to call the LoadAsync() feature and then the actual PlaySong() method as follows:

Code SoundPlayer songPlayer = new SoundPlayer();

```
try
{
    System.Console.WriteLine("Attempting Your File Now...");
    songPlayer.Stream = File.OpenRead(@"C:\Users\Naye\" + songPath);
    songPlayer.LoadAsync();
    songPlayer.PlaySync();
}
```

Following the completion of the features above our program will require an exception handling object to save from migraines, sadness, and that dreaded irritation we call debugging *hopefully*.

To allow for further understanding, a dialogue return is the best for client friendly format.

Something nonchalant to way we screwed up the file path or they did but not point blame.

It is also important to allow for the users options to be provide and map out a new linear process. I will be calling the string input for the error return as audioexcom1.

Code catch (Exception)

```
{
    Console.WriteLine("Your File Did Not Load Effectively. Please Check That It Is In WAV Format And Try Again!");
    Console.WriteLine("Please Choose 'Try Again' Or Go Back To Your Options Prompt With The Command 'Options': ");
    string audioexcom1 = Console.ReadLine();
```

Once we have received the users input in the event of an exception handle, we can move toward their desired choice.

Using an If/Else/Else If statement, we can cover the bases of the user options possibly selected as shown below:

Code if (audioexcom1 == "try again")

```
{
    Console.WriteLine("Thank You For Your Input...");
```

```

        goto AudioCat;
    }

    else if (audioexcom1 == "Options")
    {
        Console.WriteLine("Going Back To Your Options Menu...");

        goto MenuOptions;
    }
}

```

Now that this is completed the file should be able to be played in full and will handle specifically WAV file format. Try it out, and see what you can do with it. It does not have the media stop, pause, or editing options in this quick manner, so look into it and see how you will make your own implementation and what you would add for flavor.

Enjoy.

Happy Coding.

-°Koda-King°-

Dalvik - May 12, 2018, 2:56 pm

I scanned over the tutorial and first of all saw no stream disposing, but I let it slide. Then I read "goto" and now I feel so much pent-up anger and I hope I can release some by writing this post.

0xDEAD10CC - May 12, 2018, 3:35 pm

Wouldn't advise anyone to use this example code in production..

int - May 13, 2018, 5:55 am

Shitty code.

Also c:\Users? Nigga u dumb?

Environment.ExpandVars(%userprofile%)

goto? Pro c# man