Open MusikDB Standard

Revision: Alpha 0.0.1

August 6, 2022

Contents

1 Motivation

Developing software for music automation has made me realize that there is no simple music database standard commonly used. This is an attempt at making a simple database structure that can be integrated easily with existing and new software yet still provide the necessary data to the user/developer. With this standard, sufficient song data is present to be practical yet easy to handle on the back end without superfluous data that won't be used.

Keeping Data Fields minimal and relevant is at the core of this goal. Looking at other music database schemes the map of parameters is overwhelming for someone just looking to get a database up and running with their code.

Encouraging a standard is something that is the aim of this project. Many automation systems have their own format for storing music data and don't publish that standard openly. This, clearly, is a way to make migration to other platforms difficult. By providing at least core music parameters we hope to standardize the principle elements of music storage in the space.

2 Ecosystem Design

Designing an ecosystem for this standard revolves around having as minimal elements as possible which allows for more fredom and flexibility with its deployment. The three elements that compose the ecosysetm are an interpreter, Database, and Music Storage.

3 Database

The Database itself is a list of all song data and filenames of the audio files. The format of this data is discussed in the next section. The data in the Database is essentially the the metadata or the tools used to sort a song library. Use of the database information can find a genre group, or pinpoint a song based on Artist name. However, the database itself only contains half of the relevant data for a practical application you would want to play the audio file which is not stored in the database. The audio file is stored in a folder elsewhere on the system.

4 Music Storage

This ecosystem stores all audio files in one folder on the system instead of sorting songs into multiple folders based on paremeter, E.G. genre. Placing all songs in one directory is used for a handfull of reasons. With all audio in one location it makes addressing a particular song easy based on the name of the song filename. This is far easier than traversing multiple directories to get to the song location. It also means that if the location of the storage location changes there is only one location in the interpreter that needs be changed. As well, having all songs in one location means that backup and upkeep, through a network location, is a trivial task.

4.1 Interpreter

The interpreter is some code base that integrates the Database and Music Storage to provide a complete audio and metadata set to a program. The interpreter takes the files contained in the music storage location and can then use the metadata provided in the database to search efficiently the library and select the appropriate song file. We can see how having one music storage location

is useful as it is easy to point the interpreter to a different location to read audio files from. Flexibility of location means that different library's can be swapped by changing one file path name in the interpreter, a feature that makes manipulation of different library's trivial. Having only two elements to this ecosystem means that if a song needs to be added or removed from the system there are only two locations in which data needs to be modified. Having this regime also means that removing songs is particularly easy as you can delete data from both portions knowing only the information in one. For example. If a audio file is deleted the corresponding data in the database can be pointed to and removed with only file name, and visa versa if a database entry is removed the corresponding audio file can be removed automatically.

5 Standard

Data for the MusikDB standard is stored as comma separated values. This is to make reading, writing and storing the files fast and easy. This also is supported by multiple languages Serialization and Deserilization processes that can automatically cast song entries and data fields to a structure in your program. File extentions can be set to any value that is desired. The ".MDB" extention is the extention that is used by default but as the files are essecially ".CSV" that extention can also be used for ease of opening/editing in common programs.

5.1 Data Fields

Distilling all song data to the relevant parameters, as we have discussed, is paramount in this standard. The identified parameters are as follows:

The table below shows the order and the parameters present in the standard. Along with this information, the recomended type to cast the values to are also listed if a SERDE with strong types is desired.

Data Field Name	Function	Data Type
title	Name of Song	String
artist	Name of Artist	String
album	Name of Album	String
genre	Genre Name	String
year	Year song created	String
duration	Actual Song Duration	String
location	Song Creation Location	String
label	Name of Label	String
filename	Name of Audio File	String
link	Link to song	String
data	Field for extra data	String
add_date	Date added to database	String
id	Unique id for Organization	String

6 Examples of Use

6.1 Python

Coming Soon

6.2 Rust

Coming Soon