Manager

Manager has three classes song, playlist, Manager.

Playlist

Playlist has 3 instance variables.

Db_id : an integer given to it by the database

Name: a string set by the user

Songs : a list of songs that it contains given to it by the managers get _all_data method

Song

Song has 16 instance.

Db_id given by the database , path of where the song is, tag used by tiny tag to get the songs other instances that are title, album, artist, track total (the number of tracks in the album the song is in),

Duration, genre, year, composer, bitrate, sample rate, comment, image and file size.

Manager

Manager has a class variable called “SQLITE_SCHEMA” and 17 methods.

SQLITE_SCHEMA

It has 3 tables

Songs: it has 2 fields the song_id (generated by the database itself) and path.

Playlists: it has 2 fields the playlist_id (generated by the database itself) and name.

SongsPlaylistsGroups: it has 4 fields record_id (generated by the database itself), song_id(foreign key referencing the song_id field in songs), playlist_id(foreign key referencing the playlist_id field in playlists) and finally playlist_order which is the songs position in a playlist.

The __init__ method

Its arguments are the instance itself (like all the other methods) and db_path which uses it to set the instance variable db_path.

And then it sets the instance variables songs and playlists to an emty dictionary.

Then it calles the open_connection method and if that return true it calls the setup_database and get_all_data methods.

Open_connection

Tries to connect to the database and set a cursor and then it return 1.

It passes the exceptions to the show_errors_to_user.

Is_database_valid

…

Setup_database

If the database is not valid it calles the close_connection method and raises and exception.

Otherwise it sets up the database using SQLITE_SCHEMA(which contained the databases schema) and commits it and if there are any errors it passes it to show_errors_to_users plus the “setup_database” string which is where the error occurred.

Get_all_data

Gets all the song paths stored in the database plus their ids generated by the database.

Uses the path and id to make instances of songs and then populates the songs dictionary (instance variable initially set to an emty dictionary in the init method) which in it the keys are the ids given by the database and the values are instances of the song class.

Uses the name and id to make instances of playlists and then populates the playlists dictionary (instance variable initially set to an emty dictionary in the init method) which in it the keys are the ids given by the database and the values are instances of the playlist class and creates a list called playlist_ids containing all of the playlist ids in the database.

Iterates over the playlist ids in playlist_ids and gets all the song_ids of the songs it contains plus the order that they were add to the playlist and then populates each playlist instances songs variable(a list containing instances of the songs it has). If any errors occur it passes them to the show_errors_to_users method.

Add_playlist

Takes in the playlists name inserts it into the database gets the id generated by the database to populate the playlists dictionary(instance variable initially set to an emty dictionary in the init method) and commits the changes. And returns True.

If any errors occur it passes the error to the show_errors_to_users method and returns False.

Remove_playlist

It takes in either the name of the playlist the user wants to remove or its id.

If it takes the name it gets the playlists id from the playlists dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method) and then using the id it deletes the key-value of that id in the playlists dictionary plus every record that contains the id in the playlists and songsplaylistsgroups tables in the database and commits the changes and returns True.

If any errors occur it passes the error to the show_errors_to_users method.

Add_song

Takes in the songs path inserts it into the database gets the id generated by the database to populate the songs dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method) and commits the changes. And returns True.

If any errors occur it passes the error to the show_errors_to_users method and returns False.

Remove_song

It takes in either the path of the song the user wants to remove or its id.

If it takes the path it gets the playlists id from the songs dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method) and then using the id it deletes the key-value of that id in the songs dictionary plus every record that contains the id in the songs and songsplaylistsgroups tables in the database and commits the changes and returns True.

If any errors occur it passes the error to the show_errors_to_users method.

Songs_dict_filter

…

Playlists_dict_filter

…

Add_song_to_playlist

it either takes a songs path and the name of the playlist the user wants to add the song to or their ids.

If it takes the path and the name it gets the playlists-songs id from the playlists-songs dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method)

Adds the instance of song to the instance of the playlists songs list.

Inserts the song-playlist ids and the length of the instance of the playlists songs list(which is the playlists order) in the songsplaylistsgroups table in the database and commits the changes and returns true.

If any errors occur it passes the error to the show_errors_to_users method.

Remove_song_from_playlist

it either takes a songs name and the playlist the user wants to remove the song from or their ids.

If it takes the path and the name it gets the playlists-songs id from the playlists-songs dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method).

removes the instance of song from the instance of the playlists songs list.

Gets the first records(of the song being added to the playlist) playlist_order stores it in a flag deletes the record from the database containing the song-playlist id and flag and then updates the playlist_order field (decreases it by 1) of all the records that have a larger playlist_order and the commits the changes and returns true.

If any errors occur it passes them to the show_errors_to_users method and returns False.

Edit_playlist_name

It takes in the id of the playlist and its new name it changes the name of the instance of that playlist in the playlist dictionary(instance variable initially set to an emty dictionary in the init method and then populated in the get_all_data method) using the playlist id.

If any errors occur it passes them to the show_errors_to_users method.

Filter

It takes in a database query and tries to execute and then commit the changes.

If any errors occur it passes them to the show_errors_to_users method.

Close_connection

Closes the connection to the database.

Show_errors_to_users

It takes in the error message and a key-word argument place and prints the error message and the place it took place.