MyProject Documentation

Contents

Namespace `FTC-Music-Player`

Sub-modules

- FTC-Music-Player.lib
- FTC-Music-Player.main
- FTC-Music-Player.models

Module `FTC-Music-Player.lib`

Definitions of some generally useful functions to use throughout the project.

Functions

Function `TODO`

```
def TODO(
    s: str
)
```

Module `FTC-Music-Player.main`

Module `FTC-Music-Player.models`

Defines several classes modelling data that will be received from the C# backend to be used by the Python frontend:

- User – A class to represent a single User of the application.
- Content – Top level class modelling any kind of content an Artist could make.
- Album : Content – A collection of Songs.
- Playlist : Album – A /mutable/ collection of Songs created by the User.
- Song : Content – A single Song.
- Artist – An entity that has zero or more Content.

Classes

Class `Album`

```
class Album(
    name: str,
    artist,
    songs: list[FTC-Music-Player.models.Song]
)
```

Immutable collection of Songs made by the same Artist and grouped togther.

- name – Name of the Album.
- arist – Name of the Artist that made the Album.
- songs – List of Songs in the Album.

Ancestors (in MRO)

- FTC-Music-Player.models.Content

Descendants

- FTC-Music-Player.models.Playlist

Class variables

Variable `songs` Type: `list[FTC-Music-Player.models.Song]`

Class `Artist`

```
class Artist(
    name: str,
    albums: list[FTC-Music-Player.models.Album] = [],
    songs: list[FTC-Music-Player.models.Song] = []
)
```

Class variables

Variable `albums`   Type: `list[FTC-Music-Player.models.Album]`

Variable `name`   Type: `str`

Variable `songs`   Type: `list[FTC-Music-Player.models.Song]`

Class `Content`

```
class Content
```

Class modelling any form of Content made my an Artist, inherited by several other classes.

Descendants

- FTC-Music-Player.models.Album
- FTC-Music-Player.models.Song

Class variables

Variable `artist`

Variable `name`   Type: `str`

Class `Playlist`

```
class Playlist(
    name: str,
    songs: list[FTC-Music-Player.models.Song] = []
)
```

Playlist class to be used in the interface between the back- and front-ends.

- name – Name of the Playlist.

- arist – Name of the Artist that made the Playlist.

- songs – List of Songs in the Playlist.

- add_song() – Adds a Song to the list of Songs.

Ancestors (in MRO)

- FTC-Music-Player.models.Album
- FTC-Music-Player.models.Content

Methods

Method `add_song`

```
def add_song(
    self,
    song: FTC-Music-Player.models.Song
)
```

Class `Song`

```
class Song(
    name: str,
    artist
)
```

Models a single Song that can be a part of zero or more Albums or Playlists, as well as followed by zero or more Users.

Ancestors (in MRO)

- FTC-Music-Player.models.Content

Class `User`

```
class User(
    handle
)
```

A User class to model the data for a user of the application.

- username – Personal identifier
- playlists – List of playlists/albums the user has saved
- favourites – Special playlist of the user's favourite songs.
- followed – List of all the artists that the user has followed.

Initialise a User instance with no playlists, no liked/favourited songs and no followed artists.

Class variables

Variable `favourites` Type: `FTC-Music-Player.models.Playlist`

Variable `followed_albums` Type: `list[FTC-Music-Player.models.Album]`

Variable `followed_artists` Type: `list[FTC-Music-Player.models.Artist]`

Variable `playlists` Type: `list[FTC-Music-Player.models.Playlist]`

Variable `username` Type: `str`

Methods

Method `change_username`

```
def change_username(
    self,
    new_handle: str
)
```

Method `create_playlist`

```
def create_playlist(
    self,
    name: str
)
```

Method `follow_album`

```
def follow_album(
    self,
    album: FTC-Music-Player.models.Album
)
```

Method `follow_artist`

```
def follow_artist(
    self,
    artist: FTC-Music-Player.models.Artist
)
```

Method `like_song`

```
def like_song(
    self,
    song: FTC-Music-Player.models.Song
)
```

Generated by *pdoc* 0.10.0 (https://pdoc3.github.io).