

Part II.

Code from Team F

Gardon Tomas, Jaros Miroslav, Meena Rajni

The Parser

Created parser have defined datasource here:

<http://beets.readthedocs.org/en/v1.3.9/plugins/web.html?highlight=web#json-api>

Source code can be found here: https://github.com/xjaros1/Secure_coding_project/

The data structure is plain JSON which can be in two different ways:

- Items
 - Are covered in the Object where key is Items and value is Array of Object
 - { "items" : [{ItemObject},{ItemObject}...{ItemObject}] }
 - ItemObject Contains
 - i. MusicBrains data
 - ii. Name of Album, Name of item, Artist and other things
 - iii. Length of item and other meta-data
- Albums
 - Are Covered as Object where key is "albums" and value is Array of Object
 - { "albums" : [{AlbumObject}, {AlbumObject},...,{AlbumObject}] }
 - AlbumObject contains:
 - i. MusicBrains data
 - ii. Name of Album, Name of Artist
 - iii. Number of tracks, album genere and other things

Implementation Details

The parser is basicaly splited into two functionalities

- Json parser
- Data interpreter

Json parser is meant as general code capable of parsing json format (but we didn't fully succeeded)

Json parser

Json parser consist of several classes which defines both derivation tree of language and builder of this tree. List of classes:

- JValue -- Represents possible value in Json node, data types are float, int, Array, Object, String
- JArray -- Represents Array of JValues
- JObject -- Represents key: value data structure, key is always String, value is JValue
- JParser -- Represents pushdown automata. Data are inserted and processed through operator>>

Data representation

For data representation was created three classes. Two of them are only data storages: Item class and Album class. Third and last class is manager class, which contains lists of mentioned storage classes and code capable for filling them.

Item and Album:

Both classes contains easily editable list of keys and possible types of values (FLOAT, STRING, INTEGER). Its implemented as several c++ map containers, so there is no problem to add, remove or change some keys or values from data storage.

MediaManager:

Manager class, takes data from parsed Json and tries to add them to own internal memory. For every parsed data it checks key and value type with predefined ones in data store classes mention above. If everything is correct manager creates new object of Item or Album, add this object to own internal list and fill it up with data. Otherwise exception is thrown. (Data from internal memory are not printed to console, valid input is recognized only by exceptions = no exception thrown).