View module code review

Overview

View module of library-cli is based on two parent classes: View and AddDocumentView. It implements basic inheritance and a lot of static functions to achieve its goals.

Goals

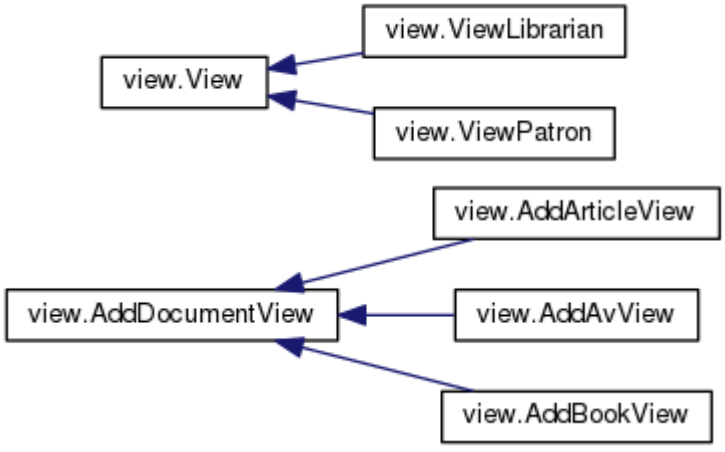
The purpose of this module is to provide functionality for interaction with clients. It is used for retrieving client data (e.g. login, password) and sending responses (e.g. displaying information about database objects).

Usage Scenarios

Since the main logic of library-cli is placed inside other modules (core.py and database.py), they need to communicate with a client somehow. This is where view module is indispensable.

One of the possible cases happens when a librarian queries information about a particular user. First, core.py calls method next_command from the class View. Then it communicates to the database, retrieves the needed information from there and calls method show_user from LibrarianView to display this information to the librarian.

Class Diagrams



Code is here

Example

1. Design decisions
    1.1. NameTTT: You should use HASH_TABLE for 'yyy' instead of ARRAY.
        1.1.1. Owner: the decision of using ARRAY is based on the fact that ARRAYs .
        1.1.2. NameTTT: That might be true, but you are compromising execution time, 'yyy' is used to perform several searches. Having a HASH_TABLE one can have constant complexity in searching
        1.1.3. Owner: Ok, I will change the design and implementation. [CLOSED]
    1.2. ...
2. API design
    2.1. ...
3. Architecture (including inheritance hierarchy, if any)
4. Implementation techniques, in particular choice of data structures and algorithms
5. Exceptions handling - Contracts
6. Programming style, names
7. Comments and documentation

Categories for review

Use your telegram alias as a name

1. Design decisions

1.1. Reviewer1: I believe in show_documents function (View class) you don't need to write empty print() command. You may add '\n' in the beginnings of other prints.

1.1.1. Owner: I am not sure if it is possible. As you can see, only loop contains a print operation (lines 68-69). So if I put '\n' in the beginning, I get a series of unnecessary new lines. Actually, I have an idea for a possible improvement and would like to discuss it. [TO BE DISCUSSED] This improvement is not feasible. [CLOSED]

1.2. Reviewer2: Variable arrow is outside any class, better to make it static variable of a class, for example, of a class View.

1.2.1. Owner: Though it will make the code a bit longer, design is above all. I will consider your advice. [CLOSED]

1.3. Reviewer3: Functions that are involved in modification of objects are not easy to use. When a librarian modifies an object and wants to leave a field as it was before (e.g. Name: fenchelfen), he or she has to retype this information again (e.g. Name: fenchelfen -> fenchelfen). To improve UX, you should add a new feature: librarian leaves a field empty not to change it.

1.3.1. Owner: Oh, this issue has been fixed recently by my teammate, this feature exists in a newer version. [FIXED]

1.4. ...

2. API design

2.1. Reviewer1: I think you should make shorter names of functions. It is not convenient to remember them all.

2.1.1. Owner: I think names should be self-explanatory. [TO BE DISCUSSED] Ok, I will make shorter function names and more detailed comments then. [TO BE IMPLEMENTED]

2.2. …

3. Architecture (including inheritance hierarchy, if any)

3.1. Reviewer1: According to the class diagram there is no inheritance between addDocumentView and View classes, but it presents in the code. You should edit one of them.

3.1.1. Owner: This is my mistake, I posted an obsolete version of the diagram. I will put a new one here as soon as possible. [FIXED]

3.2. Reviewer1: I think you should change get_password function (View class), because now a patron which has spaces in his or her password will not be able to log in.

3.2.1. Owner: Yes, split operation should be definitely removed. [CLOSED]

3.3. Reviewer1: You should receive an argument list in show function (addDocumentView class). Because now you have to delete and insert ones more the documents which were edited.

3.3.1. Owner: I cannot deny that having an attributes list in show_function would increase flexibility of the module a lot. However, this will cause a plenty of changes in other modules so this is not feasible. [CLOSED]

3.4. ...

4. Implementation techniques, in particular choice of data structures and algorithms

4.1. Reviewer4: You should use hashmap instead of dictionary,which is based on hashset because of possible null elements.

4.1.1. Owner: Dictionary is a built-in mapping data structure so it is a good choice in this case. Also, it has all of the hashmap's properties, and null elements can be easily handled for this reason. [CLOSED]

4.2. …

5. Exceptions handling - Contracts

5.1. Reviewer5: In many methods you already handling exceptions about existence of values with if/else.You can do the same, but with try/except.

5.1.1. Owner: This is a good idea. However, some of the logic is heavily dependent on these is_null branches.Thus, nothing can be changed. [CLOSED]

5.2. Reviewer5: And I think you need handling exception in show_document in class View, because of existence of key and str(val)

5.2.1. Owner: Yes, I had better do some checking whether the given value is not none. I think, I will use assertions to implement this. [CLOSED] P.S. There is no need in checking if the key is present since every document dictionary certainly has one.

5.3. …

6. Programming style, names

6.1. Reviewer5: You should inherit class object in View class (class View(object):).Because of: If the class is not inherited from any classes, explicitly inherit from the object class. For the rest it's a very well organized code.

6.1.1. Owner: As you might remember, Eiffel also has a general type that covers all the others — ANY. I know for sure that eiffel programmers do not state this fact explicitly. However, I am not familiar with python style conventions and would be glad to discuss this issue with you and other reviewers. [TO BE DISCUSSED] Your assertion is applicable to Python2 while I am using Python3. [CLOSED]

6.2. ...

7. Comments and documentation

7.1. Reviewer6: Used informal language in description of methods show_document and show_copies(prettified), show_dialog_add_user and show (Prompts). Commentary for method show_message is redundant (method name is self-explanatory) and should be removed.

7.1.1. Owner: Ok, I will try to write more formally. By the way, I do not agree that comment in show_message is redundant, but it should be definitely changed since it just repeats the function’s name. [CLOSED]

7.2. Reviewer1: May be it would be good to change the texts which will be printed in show_result_of_adding_user function. I suppose it is not quite understandable to a user to receive message like "Wrong type of user". I think he will not understand it.

7.2.1. Owner: I am sure that this is not clear from the code. But when you see these messages in the context of user addition, you can easily understand where the problem lies. [CLOSED]

7.3. Reviewer4: There is no description for most part of methods: all parameters of methods should be described.

7.3.1. Owner: I agree, it is not clear what type an argument has without comments. It will be implemented [CLOSED]

7.4. ...