Code review

Overview

DocumentController class for LMS

Goals

DocumentController controls all processes connected to documents, such as: adding, deleting and modifying documents, also displaying documents.

Usage Scenarios

If one wants to do anything with documents, everything goes through this class.

Class Diagrams

Code [REMOVED]

Categories for review

1. Design decisions

Example:

[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 are lighter than HASH_TABLES

[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]

2. *API design*
3. Architecture (including inheritance hierarchy, if any)
4. Implementation techniques, in particular choice of data structures and algorithms
5. Contracts
6. Programming style, names
7. Comments and documentation

1. Design decisions

[1.1] Reviewer1: private HashSet<Author> findAuthors may be will be better do decrease the number of conditions of type: (field!=null) by adding the function create_author() which is guarantee that the fields will be not none?

[1.1.1] Owner: the thing is, our database automatically creates an author in case there is no such author in the system. There are a lot of null conditions because there can be no such authors (in which case, the author will be automatically created), or there can be the author which has the same first name (then the author will be added based on his last name). It is a little work-around, because when we actually used some kind of create_author() there were cases when authors were added to database even though they are already there. [CLOSED]

[1.2] Reviewer2:

```
TypeDocument type =
typeDocumentService.findByTypeName(documentModel.getType().getTypeName());
```

Maybe it would be more convenient to create descendants of class Document, than have “type” as a field.

[1.2.1] Owner: database restrictions. We tried to do it the inheritance way, but we failed miserably. [CLOSED]

[1.3] Reviewer2: every type of Documents have the same services (e.g. publisherService, when Audio and Video materials have no publishers). What if some types of documents will need new services?

[1.3.1] Owner: In reality, we have services for all the entities we use. Spring framework provides us with repositories (way to represent a database table) and services (work with repositories) - that’s just the way it works best. If some types of documents will require new services (or even repositories) it is easy to write a new one since repositories extend JpaRepository<Value, Key> and are marked by @Repository tag and services are marked with @Service tag. [CLOSED]

[1.4] Reviewer3: is it possible somehow reduce code repetitions of code below?

```
ParserToken token = TokenAuthenticationService.getAuthentication(request);
if (token == null) throw new UnauthorizedException();
```

Like during handling requests check token first and then pass arguments to corresponding function. Moreover this approach gives an opportunity to check for privileges (like you do it “if (!token.role.equals("admin")) throw new AccessDeniedException();”) before passing arguments to a function.

[1.4.1] Owner: When we implement privileges it is very likely that we will implement some methods to check the access and privilege using a single line of code. But actually that could be a wee bit longer than the approach we are using right now.

[1.4.2] Reviewer3:I don’t suggest to write a function to check and then invoke it from each method which deal with requests. No. I just ask is it possible to do it BEFORE invoking method. Just imagine if you need to add 20 more functions to work with requests then you should add in every single function this lines of code. What i suggest is to modify your handling mechanism in such way that you firstly look on authorization then on librarian\admin rights if need and only after that you should pass arguments to the function which work with request and response. Consequently, there will no need to pass token as an argument to every single function because on the moment your function is invoked all work with token will be done.

[1.4.3] Owner: This is authentication which is absolutely needed to ensure that users will not be able to do some forbidden things. On the back end, this is all we can do. Of course, the problem could be solved on the front end of the program by not showing the corresponding buttons. As for right now, we are not planning to rewrite the authentication process. [CLOSED]

[1.5] Reviewer4:

```
private HashSet<Author> findAuthors(Set<Author> setAuthors)
Set<Author> authors = findAuthors(documentModel.getAuthors());
```

Is there any reason why you use HashSet as returning value of method but, when you use this method, you get the result in the variable with type Set? Maybe it will be better to use the same types.
[1.5.1] Owner: Set<> is an interface which is used by many classes (e.g. HashSet<>). By passing Set<> as an argument we can ensure that we can pass any class implementation of Set<> without getting IllegalArgumentException.
[1.5.2] Reviewer4: I mean why method findAuthors gives back HashSet<> while the type of variable 'authors' is usual Set<>. In this case you cannot use methods which belong only to HashSet<> type, because the type of authors is Set<>.
[1.5.3] Owner: the only method unique to HashSet<> is clone(), and that is the method we are definitely not going to use at any time. In addition to that, as I already mentioned, using Set<> as class variable increases possible future flexibility which was kept in mind when designing it. [CLOSED]

2. API design

[2.1] Reviewer1: the purpose of findAuthors method seems not so useful for me, because we are giving back an hashtable of totally different information about authors: id, first name, last name. What about adding constructor of the class that initializes an instance, object of class Author and giving back a hashmap of objects of class author or at least their id's?
[2.1.1] Owner: Method findAuthors returns HashSet<Author> because our documents contain their authors in Set<>. So, we are not passing unnecessary information - we are passing Authors as objects of a class Author [CLOSED]

3. Architecture (including inheritance hierarchy, if any)

[3.1] Reviewer1: [2.1] - may be will be useful to add a class Authors?
[3.1.1] Owner: we already have it. [CLOSED]

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

[4.1] Reviewer2: method findAuthors() may find author with requested first name even if his second name differs from requested one, but author with not right first name end right second name will not found. Example: If request is id=null, firstName=John, secondName=Smith. Author1 =John Way, author2 = Bill Smith, It finds only author1.
[4.1.1] Owner: As the method is used only by documentController (which is primarily used for adding and modifying documents), the only times it is called are when someone wants to modify a document or add a new one. Because, as we believe, these operations are performed by librarians with exact books they work with in their hands, there is little to no possibilities of mistaken input. Also, our database is set up in such a way that John Smith would be automatically added to the database. [CLOSED]

5. Exceptions handling - Contracts

[5.1] Reviewer5: you should throw null reference exception if any input parameter in the constructor is null.

[5.1.1] Owner: please emphasize what exactly you are referencing to. For example, due to database restrictions, all types of documents have all fields (e.g. AV materials have Publishers), but unnecessary ones are null.
[5.1.2] Reviewer5: public DocumentController(DocumentService documentService, TypeDocumentService typeDocumentService, AuthorService authorService, PublisherService publisherService)
I am saying about this part of code, may this input parameters be null?
[5.1.3] Owner: None of them can't be null as services and repositories are created and connected on server startup automatically. [CLOSED]

[5.2] Reviewer4:

```
if (documentModel.getPublisher().getId() != null) {
        publisher = publisherService.findById(documentModel.getPublisher().getId());
} else {
        publisher =
        publisherService.findByPublisherName(documentModel.getPublisher().getPublish
        erName());
}
```

In the case when both Id and Name of publisher are null the publisher will become null too. And you don't handle this situation.
[5.2.1] Owner: The situation is not handled because all our types of documents (Books, AV materials, Journal articles and what not) are all represented as instances of class Document. Since some of them may not have publishers, the situation is never handled due to unnecessity. [CLOSED]

6. Programming style, names

[6.1] Reviewer6: I see you have a method findById(), but then why you don't have findByName() instead of findByPublisherName()?
[6.1.1] Owner: At some point we refactored the code and thought that having different name for basically same methods would help us to navigate through code. It's much easier to find method __findByPublisherName()__ than __findByName()__ when there are multiple services using it. [CLOSED]
[6.2] Reviewer5: in method findAuthors you can write:

```
if (setAuthors == null)
        return authors;
for (Author author : setAuthors) {
…
```

It will decrease nesting of code.
[6.2.1] Owner: good idea, will fix [TO-BE-IMPLEMENTED]
[6.3] Reviewer5: Document findDocument = documentService.findById(id); - don't name instances starting from verb.
[6.3.1] Owner: good call, will fix [TO-BE-IMPLEMENTED]

[6.4] Reviewer5: private HashSet<Author> findAuthors(Set<Author> setAuthors) - here you should rename input parameter setAuthors to authorsSet for better understanding, because now "set" seems as a verb.
[6.4.1] Owner: will fix [TO-BE-IMPLEMENTED]

[6.5] Reviewer4:

```
public void removeDocumentId
```

It seems like you remove Id from document but the document itself will be not removed. So it is better to rename method as `removeDocumentById`.

[6.5.1] Owner: good, will fix [TO-BE-IMPLEMENTED]

[6.6] Reviewer4:

```
if (document == null)
        throw new UserNotFoundException();
```

I think here should be `DocumentNotFoundException()` instead of `UserNotFoundException()`

[6.6.1] Owner: good call, will change [TO-BE-IMPLEMENTED]

7. Comments and documentation

[7.1] Reviewer7: I suppose it will be better to have some comments (docs) on all your methods, that will briefly describe the main goals and purpose of such implementation.

[7.1.1] Owner: good call, will fix. We actually need to do this for the entire project, so that may take a while. [TO-BE-IMPLEMENTED]