Week 16 – Design Patterns

blog.bookly.online

alex May 22, 2020

This week we looked at software patterns. We should integrate one of them into our code. Patterns like MVC, Observer and Singleton were not allowed because most frameworks offer them “for free”.

We decided to implement the ***SOFA*** principle for our FriendshipBookService. It expects short methods with few argument and one layer of abstraction. Each method is suppose to do one thing.

This principle caused great controversy in our group. On the one hand it doesn’t seem worth the effort because the project is too small to really call for a design pattern. In a small project the extraction of functionality does not improve the code readability a great deal. It might even hinder the reader of our code because the functionality of each method has to be determined and you have to jump from place to place in code. (Even if you can do so easily with the help of your IDE, it’s annoying.) We tried to make this the least painful possible by using “meaningful method names” that make their functionality clear. On the other hand, the pattern helps in a large project to get a quicker understanding of the code. It forces coders to make a habit of thinking about the various functionalities that they are using and structuring them in a coherent manner. Our consensus is that as long as you don’t overuse it. It is a useful tool. We will keep the pattern in our code and see if we will be able to use it more meaningfully in the future.

Update

It was pointed out that SOFA is a design principle and not a design pattern, so there was a slight misunderstanding on our side. We decided to go with the Null Object Design Pattern. The intent of a Null Object is to encapsulate the absence of an object by providing a substitutable alternative that offers suitable default do nothing behavior. As we have various pictures and stickers, we decided to use the Null Object Design Pattern to implement our DummyImage. It allows us the abstract handling of null away from the client. Apart from that the refactoring enables us to get rid of some code duplication, which was of interest because of the metrics topic.

Here you can see our code:

Before:

```
public class FriendshipBook {

  ...

    @JsonIgnore
    @OneToOne(cascade = {CascadeType.ALL})
    @ApiModelProperty(notes = "the cover image of the book",
            position = 3)
    private FriendshipBookCover cover;

    @OneToOne
    @ApiModelProperty(notes = "the user of which the book belongs to. is extracted from the
authentication",
            position = 4)
    private User user;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, targetEntity =
Page.class)
    @ApiModelProperty(notes = "list of all pages of the book",
            position = 5)
    private List<Page> pages = new ArrayList<>();

    @ApiModelProperty(notes = "ID of the Theme template",
            position = 6)
    private int theme;

    @JsonIgnore
    @OneToOne(cascade = {CascadeType.ALL})
    @ApiModelProperty(notes = "the first sticker of the book",
            position = 7)
    private FriendshipBookSticker sticker1;

    @JsonIgnore
    @OneToOne(cascade = {CascadeType.ALL})
    @ApiModelProperty(notes = "the second sticker of the book",
            position = 8)
    private FriendshipBookSticker sticker2;
```

After:

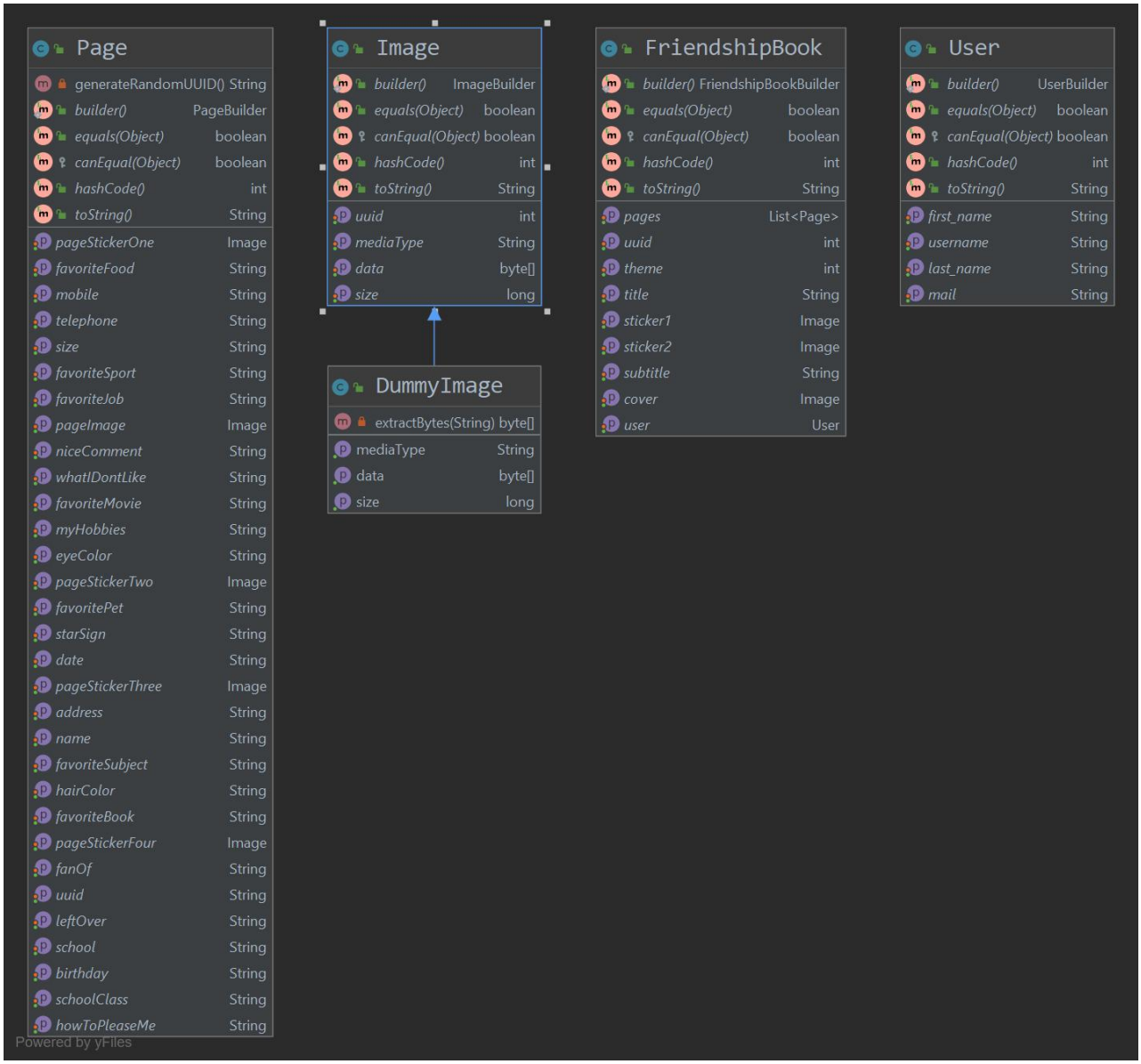
```
public class FriendshipBook {

...

    @JsonIgnore
    @OneToOne(cascade = { CascadeType.ALL })
    @ApiModelProperty(notes = "the cover image of the book", position = 3)
    private Image cover = new DummyImage();

    @OneToOne
    @ApiModelProperty(notes = "the user of which the book belongs to. is extracted from the
authentication", position = 4)
    private User user;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, targetEntity =
Page.class)
    @ApiModelProperty(notes = "list of all pages of the book", position = 5)
    private List<Page> pages = new ArrayList<>();

    @ApiModelProperty(notes = "ID of the Theme template", position = 6)
    private int theme;

    @JsonIgnore
    @OneToOne(cascade = { CascadeType.ALL })
    @ApiModelProperty(notes = "the first sticker of the book", position = 7)
    private Image sticker1 = new DummyImage();

    @JsonIgnore
    @OneToOne(cascade = { CascadeType.ALL })
    @ApiModelProperty(notes = "the second sticker of the book", position = 8)
    private Image sticker2 = new DummyImage();
```

Class diagram

Before


Page
uuid String
pageImage PageImage
name String
address String
telephone String
mobile String
schoolClass String
school String
size String
hairColor String
eyeColor String
birthday String
starSign String
favoriteSubject String
favoritePet String
howToPleaseMe String
whatIDontLike String
favoriteJob String
myHobbies String
fanOf String
favoriteMovie String
favoriteSport String
favoriteBook String
favoriteFood String
niceComment String
date String
leftOver String
pageStickerOne PageSticker
pageStickerTwo PageSticker
pageStickerThree PageSticker
pageStickerFour PageSticker
generateRandomUUID() String
builder() PageBuilder
getUuid() String
getPageImage() PageImage
getName() String
getAddress() String
getTelephone() String
getMobile() String
getSchoolClass() String
getSchool() String
getSize() String
getHairColor() String
getEyeColor() String
getBirthday() String
getStarSign() String
getFavoriteSubject() String
getFavoritePet() String
getHowToPleaseMe() String
getWhatIDontLike() String
getFavoriteJob() String
getMyHobbies() String
getFanOf() String
getFavoriteMovie() String
getFavoriteSport() String
getFavoriteBook() String
getFavoriteFood() String
getNiceComment() String
getDate() String
getLeftOver() String
getPageStickerOne() PageSticker
getPageStickerTwo() PageSticker
getPageStickerThree() PageSticker
getPageStickerFour() PageSticker
setUuid(String) void
setPageImage(PageImage) void
setName(String) void
setAddress(String) void
setTelephone(String) void
setMobile(String) void
setSchoolClass(String) void
setSchool(String) void
setSize(String) void
setHairColor(String) void
setEyeColor(String) void
setBirthday(String) void
setStarSign(String) void
setFavoriteSubject(String) void
setFavoritePet(String) void
setHowToPleaseMe(String) void
setWhatIDontLike(String) void
setFavoriteJob(String) void
setMyHobbies(String) void
setFanOf(String) void
setFavoriteMovie(String) void
setFavoriteSport(String) void
setFavoriteBook(String) void
setFavoriteFood(String) void
setNiceComment(String) void
setDate(String) void
setLeftOver(String) void
setPageStickerOne(PageSticker) void
setPageStickerTwo(PageSticker) void
setPageStickerThree(PageSticker) void
setPageStickerFour(PageSticker) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
FriendshipBook
uuid int
title String
subtitle String
cover FriendshipBookCover
user User
pages List<Page>
theme int
sticker1 FriendshipBookSticker
sticker2 FriendshipBookSticker
setTheme(int) void
setPages(ArrayList<Page>) void
builder() FriendshipBookBuilder
getUuid() int
getTitle() String
getSubtitle() String
getCover() FriendshipBookCover
getUser() User
getPages() List<Page>
getTheme() int
getSticker1() FriendshipBookSticker
getSticker2() FriendshipBookSticker
setUuid(int) void
setTitle(String) void
setSubtitle(String) void
setCover(FriendshipBookCover) void
setUser(User) void
setSticker1(FriendshipBookSticker) void
setSticker2(FriendshipBookSticker) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
FriendshipBookSticker
uuid int
data byte[]
size long
mediaType String
builder() FriendshipBookStickerBuilder
getUuid() int
getData() byte[]
getSize() long
getMediaType() String
setUuid(int) void
setData(byte[]) void
setSize(long) void
setMediaType(String) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
FriendshipBookCover
uuid int
data byte[]
size long
mediaType String
builder() FriendshipBookCoverBuilder
getUuid() int
getData() byte[]
getSize() long
getMediaType() String
setUuid(int) void
setData(byte[]) void
setSize(long) void
setMediaType(String) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
PageSticker
uuid int
data byte[]
size long
mediaType String
builder() PageStickerBuilder
getUuid() int
getData() byte[]
getSize() long
getMediaType() String
setUuid(int) void
setData(byte[]) void
setSize(long) void
setMediaType(String) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
PageImage
uuid int
data byte[]
size long
mediaType String
builder() PageImageBuilder
getUuid() int
getData() byte[]
getSize() long
getMediaType() String
setUuid(int) void
setData(byte[]) void
setSize(long) void
setMediaType(String) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
User
username String
mail String
first_name String
last_name String
builder() UserBuilder
getUsername() String
getMail() String
getFirst_name() String
getLast_name() String
setUsername(String) void
setMail(String) void
setFirst_name(String) void
setLast_name(String) void
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String

After



Overall class diagram

Overall Dependencies

In our overall class diagram, we highlighted where the pattern appears.