НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Рефакторинг баз данных и приложений

Лабораторная работа № 1

Выполнил студент

Демичев Даниил Дмитриевич

Группа № P34112

Преподаватель: Логинов Иван Павлович

г. Санкт-Петербург

2021

Задание:

Для выбранного проекта произвести рефакторинг пятью методами

Проект: https://github.com/xxPFFxx/BLPS-1

Форк: https://github.com/xxPFFxx/BLPS-1/tree/refactoring

Рефакторинги:

- Длинный список параметров
- Комментарии
- Удаление неиспользуемого кода
- Переименование метода
- Извлечение метода

Длинный список параметров

Длинный список параметров усложняет чтение и понимание метода, часто его можно заменить объектом и передавать его в метод.

```
public ResponseEntity<?> uploadVideoInfo(@RequestParam String videoName, @RequestParam String videoDesc,
                                         @RequestParam String category, @RequestParam String releaseTime,
                                         @RequestParam String releaseDate, @RequestParam String link, Principal principal){
    VideoInfo videoInfo = new VideoInfo(videoName, videoDesc, category, releaseTime, releaseDate, link);
    if (videoInfoService.checkVideoInfo(link)){
        return new ResponseEntity<>(videoInfoService.updateVideoInfo(videoName, videoDesc, category, releaseTime, releaseDate, link),HttpStatus.OK);
        return new ResponseEntity<>(videoInfoService.updateVideoInfo(videoInfo),HttpStatus.OK);
    }
    else {
        return new ResponseEntity<>("Видео не найдено", HttpStatus.BAD_REQUEST);
```

В двух методах передавалось много параметров, которые являлись свойствами класса VideoInfo, поэтому можно создать заранее объект, передать его и дальше из этого объекта получать доступ к необходимым свойствам.

Комментарии

Код без комментариев становится сложным для восприятия другими людьми и даже для себя через какое-то время, так как в некоторых местах можно писать какие-то специфичные вещи

```
          this.videoInfoService = videoInfoService;
      }

+     /*
+     Каждую минуту (cron = "0 * * * * *") проверяет, не стало ли какое-то видео популярным (набрало хотя бы 10 просмотров),
+     и обновляет статус для видео, выполнивших это условие
+      */
      @Scheduled(cron = "0 * * * * *")
-     public void reportCurrentTime() {
+     public void updateStatusOnVideos() {
          videoInfoService.setPopularStatus();
  //        log.info("The time is now {}", dateFormat.format(new Date()));
      }
```

Тут, например, можно указать, что означает это cron-выражение, чтобы через какое-то время не было необходимости искать эту информацию заново.

Удаление неиспользуемого кода

Лишний и неиспользуемый код не несет никакой смысловой нагрузки, это могут быть, например, прошлые попытки сделать какую-то вещь.

```
package com.example.uploadingfiles.util;

import com.example.uploadingfiles.services.VideoInfoService;
- import org.slf4j.Logger;
- import org.slf4j.LoggerFactory;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

- import java.text.SimpleDateFormat;

@Component
public class ScheduledTasks {
    private final VideoInfoService videoInfoService;
-     private static final Logger log = LoggerFactory.getLogger(ScheduledTasks.class);
-
-     private static final SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");

    public ScheduledTasks(VideoInfoService videoInfoService) {
        this.videoInfoService = videoInfoService;
@@ -26,6 +20,5 @@ public ScheduledTasks(VideoInfoService videoInfoService) {
    @Scheduled(cron = "0 * * * * *")
    public void updateStatusOnVideos() {
        videoInfoService.setPopularStatus();
- //        log.info("The time is now {}", dateFormat.format(new Date()));
    }
}
```

Так можно удалить неиспользуемые переменные, ненужные импорты и промежуточные выводы, такие как log.info() или System.out.println(), тем самым уменьшив объем кода и немного ускорив программу.

Переименование метода

Методы должны иметь краткое однозначное название, чтобы без изучения их внутренностей можно было понять, что этот метод делает.

```
    @Scheduled(cron = "0 * * * * *")
    public void reportCurrentTime() {
    public void updateStatusOnVideos() {
        videoInfoService.setPopularStatus();
//        log.info("The time is now {}", dateFormat.format(new Date()));
    }
```

Например, при копировании откуда-то могло остаться старое название метода, в таких случаях стоит максимально упростить понимание переименованием метода.

Извлечение метода

Каждый метод должен выполнять свою конкретную задачу, и когда в одном методе разбросаны реализации разных вещей, это становится сложнее поддерживать.

```
    public ResponseEntity<?> handleFileUpload(Principal principal, @RequestParam("file") MultipartFile file
                                                                ) throws IOException {
                storageService.store(file);
-               int leftLimit = 97; // letter 'a'
-               int rightLimit = 122; // letter 'z'
-               int targetStringLength = 10;
-               Random random = new Random();
-               String generatedString = random.ints(leftLimit, rightLimit + 1)
-                               .limit(targetStringLength)
-                               .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
-                               .toString();
+               String generatedString = generateLink(10);
                //TODO проверка, нет ли такой линки уже

                return new ResponseEntity<>(videoInfoService.saveVideoInfo(null, null, null, null, null, generatedString, principal.getName()),HttpStatus.OK);
        }

+   public String generateLink(int length){
+           int leftLimit = 97; // letter 'a'
+           int rightLimit = 122; // letter 'z'
+           Random random = new Random();
+           String generatedString = random.ints(leftLimit, rightLimit + 1)
+                           .limit(length)
+                           .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
+                           .toString();
+           return generatedString;
+   }
```

Тут, например, в методе, который принимает и сохраняет файл, была реализация метода генерации случайной ссылки, поэтому логично извлечь этот код в отдельный метод.

Вывод:

В ходе выполнения этой лабораторной работы я познакомился с возможными способами рефакторинга кода, в каких случаях их стоит и не стоит применять, и на практике попробовал применить несколько из них.