Практика, неблокирующий сервер

Юрий Литвинов
yurii.litvinov@gmail.com

06.04.2017г

Разбор контрольной

Правильное (в каком-то смысле) решение, однопоточный вариант

```
public @NotNull byte[] hash(@NotNull Path path) throws
        NoSuchAlgorithmException, IOException {
    MessageDigest md = MessageDigest.getInstance("MD5");
    if (Files.isDirectory(path)) {
        md.update(path.getFileName().toString().getBytes());
        for (Path subPath : Files.walk(path).filter(Files::isRegularFile).collect(Collectors.toList())) {
            md.update(hash(subPath));
        }
        return md.digest();
    }

    try (InputStream rawStream = Files.newInputStream(path)) {
        DigestInputStream stream = new DigestInputStream(rawStream, md);
        byte[] buffer = new byte[BUFFER_SIZE];
        while (stream.read(buffer) != -1)
        {
        }
        return stream.getMessageDigest().digest();
    }
}
```

Fork/Join-вариант

```
MessageDigest md = MessageDigest.getInstance("MD5");
if (Files.isDirectory(path)) {
    md.update(path.getFileName().toString().getBytes());
    List<ForkJoinHasher> subTasks = new LinkedList<>();
    for (Path subPath : Files.walk(path).filter(Files::isRegularFile).collect(Collectors.toList())) {
        ForkJoinHasher task = new ForkJoinHasher(subPath);
        task.fork();
        subTasks.add(task);
    }
    for (ForkJoinHasher task : subTasks) {
        md.update(task.join());
    }
    return md.digest();
} else {
    try (InputStream rawStream = Files.newInputStream(path)) {
        DigestInputStream stream = new DigestInputStream(rawStream, md);
        byte[] buffer = new byte[BUFFER_SIZE];
        while (stream.read(buffer) != -1)
        {
        }
        return stream.getMessageDigest().digest();
    }
}
```

Частые проблемы

- Невнимательное чтение условия
- Недостаток тестов
- Суровый копипаст в тестах
- Конечно же, комментарии
- Излишнее доверие пользователю в main-е
- Нетехнологичность
  - Отсутствие аннотаций @NotNull/@Nullable
  - @Rule TemporaryFolder
  - try-with-resources
  - assertArrayEquals
    - Hamcrest? Только одно решение, seriously?
  - **Tools are everything!**

Домашка, Simple FTP

Требуется реализовать сервер, обрабатывающий два запроса.

- **list** — листинг файлов в директории на сервере
- **get** — скачивание файла с сервера

И клиент, позволяющий исполнять указанные запросы.

List

Формат запроса: *<1: Int> <path: String>*

- path — путь к директории

Формат ответа: *<size: Int> (<name: String> <is_dir: Boolean>)**

- size — количество файлов и папок в директории
- name — название файла или папки
- is_dir — флаг, принимающий значение True для директорий

Если директории не существует, сервер посылает ответ с size = 0

Get

Формат запроса: *<2: Int> <path: String>*

- path — путь к файлу

Формат ответа: *<size: Long> <content: Bytes>*

- size — размер файла
- content — его содержимое

Если файла не существует, сервер посылает ответ с size = 0

Примечания

- Разрешается использовать библиотеки для упрощения ввода-вывода
- Рекомендуется взглянуть на DataInputStream и DataOutputStream
- Рекомендуется задуматься об интерфейсе сервера и клиента, возможно стоит сделать что-то подобное:
  - Server: start/stop
  - Client: connect/disconnect/executeList/executeGet

Срок: **19.04.2017 23:59**.

Задача на пару, Non-blocking server

- Сервер сразу начинает отдавать клиенту содержимое одного файла, после чего разрывает соединение
- Содержимое файла можно перед началом работы прочитать в память
- Работа с клиентскими сокетами и открытие новых должно происходить в неблокирующем режиме
- Сервер должен работать в одном потоке
- Рекомендуется для клиентских сокетов включать tcpNoDelay