vikki-ya UDP_TXT ... on Jan 27 History

..

README.md 2 months ago

Домашнее задание к занятию «1.1. HTTP и современный Web»

В качестве решения пришлите ссылки на ваш GitHub-проект в личном кабинете студента на сайте netology.ru.

Важная информация

README.md

2. Если у вас что-то не получилось, тогда оформляйте Issue по установленным правилам.
3. Все задачи нужно решать в **одном** репозитории.

Как сдавать домашнее задание

1. Создайте на вашем компьютере Maven-проект.
2. Инициализируйте в нём пустой Git-репозиторий.
3. Добавьте в него готовый файл .gitignore.
4. Добавьте в этот же каталог остальные необходимые файлы.
5. Сделайте необходимые коммиты.
6. Создайте публичный репозиторий на GitHub и свяжите свой локальный репозиторий с удалённым.
7. Сделайте пуш: удостоверьтесь, что ваш код появился на GitHub.
8. Ссылку на ваш проект отправьте в личном кабинете на сайте netology.ru.

Refactoring & MultiThreading

Легенда

Достаточно часто после того, как прототип проверен (речь о том, что было реализовано на лекции), возникает задача — привести это в должный вид: выделить классы, методы, обеспечить нужную функциональность.

Задача

Необходимо отрефакторить код, рассмотренный на лекции, и применить все знания, которые у вас есть:

1. Выделить класс `Server` с методами для:
    - запуска;
    - обработки конкретного подключения.
2. Реализовать обработку подключений с помощью `ThreadPool` — выделите фиксированный на 64 потока, и каждое подключение обрабатывайте в потоке из пула.

Так как вы главный архитектор и проектировщик этого класса, то все архитектурные решения принимаете вы, поэтому будьте готовы к критике со стороны проверяющих.

Результат

В качестве результата пришлите ссылку на ваш проект на GitHub в личном кабинете студента на сайте netology.ru.

Handlers* (задача со звёздочкой)

Это необязательная задача, её выполнение не влияет на получение зачёта.

Легенда

Сервер, который вы написали в предыдущей задаче, пока не расширяемый, и его нельзя переиспользовать, т. к. код обработки зашит прямо внутрь сервера.

Попробуйте сделать его полезнее — чтобы в сервер можно было добавлять обработчиков на определённые шаблоны путей.

Это значит, нужно, чтобы можно было сделать так:

```
public class Main {
    public static void main(String[] args){
      final var server = new Server();
      // код инициализации сервера (из вашего предыдущего ДЗ)

      // добавление хендлеров (обработчиков)
```

```
        server.addHandler("GET", "/messages", new Handler() {
          public void handle(Request request, BufferedOutputStream responseStream) {
            // TODO: handlers code
          }
        });
        server.addHandler("POST", "/messages", new Handler() {
          public void handle(Request request, BufferedOutputStream responseStream) {
            // TODO: handlers code
          }
        });

        server.listen(9999);
      }
  }
```

В итоге на запрос типа GET на путь "/messages" будет вызван первый обработчик, а на запрос типа POST и путь "/messages" будет вызван второй.

Как вы видите, `Handler` — функциональный интерфейс всего с одним методом. Он может быть заменён на lambda.

`Request` — это класс, который проектируете вы сами. Для нас важно, чтобы он содержал:

- метод запроса, потому что на разные методы можно назначить один и тот же хендлер;
- заголовки запроса;
- тело запроса, если есть.

`BufferedOutputStream` берётся путём заворачивания `OutputStream` `socket`: `new BufferedOutputStream(socket.getOutputStream())`.

Задача

Реализуйте требования, указанные в легенде.

▶ Подсказки по реализации.

Результат

Реализуйте новую функциональность в ветке `feature/handlers` вашего репозитория из домашнего задания 1 и откройте Pull Request.

Так как вы главный архитектор и проектировщик этого, уже более функционального решения, то все архитектурные решения принимать вам, поэтому будьте готовы к критике со стороны проверяющих.

В качестве решения пришлите ссылку на ваш Pull Request на GitHub в личном кабинете студента на сайте netology.ru.

После того, как домашнее задание будет принято, сделайте `merge` для Pull Request.

Give feedback