15. Создание простого HTTP-сервера с помощью Java.

Теоретическое введение

HTTP (протокол передачи гипертекста) — это основной протокол взаимодействия в сети интернет, позволяющий клиентам (например, браузерам) общаться с серверами. HTTP использует модель запрос-ответ: клиент отправляет запрос, а сервер отвечает. Рассмотрим основные компоненты HTTP-сервера.

— **TCP-сокеты**: HTTP протокол функционирует на основе TCP-соединений. Для работы с сокетами в Java используется класс ServerSocket.

— **HTTP-запросы**: HTTP-запрос состоит из метода (например, GET, POST), пути (урл) и HTTP-заголовков.

— **HTTP-ответы**: Сервер должен отправить клиенту ответ, который включает код статуса (200, 404 и т.д.), HTTP-заголовки и тело ответа (например, HTML).

Java предоставляет широкие возможности для реализации HTTP-серверов. Ниже будет приведен листинг HTTP-сервера на Java. HTTP-сервер будет реализовывать функционал заметок. На сервере будет реализованы функции добавления заметок, удаления последней заметки, а также производится проверка на удаление несуществующей заметки. Пример кода приведен на Листинге 15.1.

Листинг 15.1 – Пример реализации HTTP-сервера на Java для работы с заметками

```
import java.io.*;
import java.net.*;
import java.util.*;


public class NoteHttpServer {
    private static final int PORT = 8080;
    private static final List<String> notes = new ArrayList<>();


    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(PORT))
{
            System.out.println("Note HTTP Server запущен на порту
" + PORT);
```

```
            while (true) {
                try (Socket clientSocket = serverSocket.accept()) {
                    handleClient(clientSocket);
                }
            }
        } catch (IOException e) {
            System.err.println("Ошибка запуска сервера: " + e.getMessage());
        }
    }

    private static void handleClient(Socket clientSocket) throws IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream());

        String line = in.readLine();
        if (line == null) return;

        String[] requestParts = line.split(" ");
        String method = requestParts[0];
        String path = requestParts[1];

        if (method.equals("GET") && path.equals("/notes")) {
            handleGetNotes(out);
        } else if (method.equals("POST") && path.startsWith("/add-note")) {
            String note = extractBody(in);
            handleAddNote(note, out);
        } else if (method.equals("POST") && path.startsWith("/delete-note")) {
            handleDeleteLastNote(out);
        } else {
```

```
            handleNotFound(out);
        }

        out.flush();
    }

    private static void handleGetNotes(PrintWriter out) {
        StringBuilder response = new StringBuilder();
        response.append("<html><body><h1>Notes</h1><ul>");
        for (String note : notes) {

response.append("<li>").append(note).append("</li>");
        }
        response.append("</ul>");
        response.append("<form    method='POST'    action='/add-
note'>")
                .append("<input      type='text'      name='note'
placeholder='Add note' required>")
                .append("<button type='submit'>Add</button>")
                .append("</form>");
        response.append("<form   method='POST'   action='/delete-
note'>")
                .append("<button    type='submit'>Delete    Last
Note</button>")
                .append("</form>");
        response.append("</body></html>");
        sendHttpResponse(out, 200, response.toString());
    }

    private  static  void  handleAddNote(String  note,  PrintWriter
out) {
        if (note != null && !note.isEmpty()) {
            notes.add(note);
            sendHttpRedirect(out, "/notes");
        } else {
```

```
            sendHttpResponse(out, 400, "<html><body><h1>Invalid
note</h1><a href='/notes'>Back to Notes</a></body></html>");
        }
    }

    private static void handleDeleteLastNote(PrintWriter out) {
        if (!notes.isEmpty()) {
            notes.remove(notes.size() - 1);
            sendHttpRedirect(out, "/notes");
        } else {
            sendHttpResponse(out, 400, "<html><body><h1>No notes
to delete</h1><a href='/notes'>Back to Notes</a></body></html>");
        }
    }

    private static void handleNotFound(PrintWriter out) {
        sendHttpResponse(out, 404, "<html><body><h1>404 Not
Found</h1></body></html>");
    }

    private static String extractBody(BufferedReader in) throws
IOException {
        StringBuilder body = new StringBuilder();
        String line;
        while (!(line = in.readLine()).isEmpty()) {
            // Считываем заголовки
        }
        while (in.ready() && (line = in.readLine()) != null) {
            body.append(line);
        }
        String bodyString = body.toString();
        String[] parts = bodyString.split("=");
        return parts.length > 1 ? parts[1].replace("+", " ") :
null;
    }
```

```
    private static void sendHttpResponse(PrintWriter out, int
statusCode, String body) {
        out.println("HTTP/1.1 " + statusCode + " OK");
        out.println("Content-Type: text/html");
        out.println("Content-Length: " + body.length());
        out.println();
        out.println(body);
    }

    private static void sendHttpRedirect(PrintWriter out, String
location) {
        out.println("HTTP/1.1 302 Found");
        out.println("Location: " + location);
        out.println("Content-Length: 0");
        out.println();
    }
}
```

Для запуска сервера нужно скомпилировать код в IDE. Далее в адресной строке браузера пропишите http://localhost:8080/notes

Пример сервера приведен на рисунке 15.1.

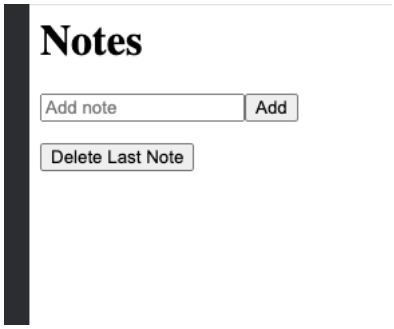

Рисунок 15.1 – HTTP-сервер на Java

Скопируйте код, скомпилируйте его и просмотрите доступный функционал.

Практическое задание

В данной практической работе представлено 3 варианта заданий. Выбор варианта осуществляется в соответствии с порядковым номером студента в списке. Если порядковый номер студента равен 1, выполняется задание варианта №1; если порядковый номер равен 3 — выполняется вариант №3. В случае, если порядковый номер превышает количество вариантов (например, 4 или более), задание выбирается циклически, начиная с варианта №1.

Также, при реализации практической работы порт, на котором будет находится сервер должен быть двойным порядковым номером по списку. Например, если порядковый номер студента по списку 1, то порт должен быть 11, если порядковый номер 10, то порт 1010 и так далее.

Также на сервере должно быть прописано ФИО студента и его шифр.

Вариант №1

HTTP-сервер с калькулятором

Задача: создать HTTP-сервер, который позволяет клиенту решать арифметические примеры.

Функционал:

— Получать от клиента числа и операцию (+, -, *, /).
— Вычислять результат и возвращать клиенту.
— При ошибке (например, деление на 0) возвращать ошибку HTTP 400.

Пример вызова:

URL: http://localhost:порт/calculate?a=5&b=3&op=+

Ответ: 8

Вариант 2

HTTP-сервер с конвертером единиц измерения

Задача: создать HTTP-сервер, который конвертирует единицы измерения (длины, веса, температуры).

Функционал:

— Получать единицы измерения (например, метры) и желаемый формат (километры).
— Преобразовывать значения и возвращать клиенту результат.
— Ошибки (например, некорректный формат) обрабатываются ответом HTTP 400.

Пример вызова:

URL: http://localhost:порт/convert?value=100&from=meters&to=kilometers

Ответ: 0.1

Вариант №3

HTTP-сервер с генератором случайных чисел

Задача: создать HTTP-сервер, который генерирует случайные числа в указанном диапазоне.

Функционал:

- Получать от клиента параметры диапазона (min и max).
- Генерировать случайное число в указанном диапазоне.
- Возвращать сгенерированное число клиенту.
- В случае ошибок (например, min > max) возвращать HTTP 400.

Пример вызова:

URL: http://localhost:порт/random?min=10&max=100

Ответ: 42