|  |
| --- |
| https://lh6.googleusercontent.com/QcftzNtI05T0Y6fjdSh1Rr2rt8oqZ1IvnLvbn1jLJ7CCyteVir3k-xBLv4SL1wAgWJsRhmmJSR0UW-RP63_GQenE4vVWv05BRoZTsmIcBccVTnfxwmsnNMvjg599x9SqZd8E3dkd |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования«МИРЭА - Российский технологический университет»РТУ МИРЭА |

Институт Информационных Технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИППО)

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7** | |
| **по дисциплине** | |
| «Разработка клиент-серверных приложений» | |
|  | |
| Выполнил студент группы ИКБО-12-18 | Копотов М.А. |
| Принял ассистент кафедры ИППО | Строганкова Н.В. |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись руководителя)* |

Москва 2020

**Содержание**

[1. Цель практической работы 3](#_Toc528748841)

[2. Задание 3](#_Toc528748842)

[3. Поэтапное выполнение практической работы 3](#_Toc528748843)

[5. Код программы 4](#_Toc528748844)

[6. Результат работы 4](#_Toc528748845)

[7. Вывод 4](#_Toc528748846)

[Список использованных источников 5](#_Toc528748847)

# 1 Цель практической работы

Знакомство с протоколом передачи файлов прикладного уровня HTTP, а также научиться работать с GET- и POST-запросами.

# Задание

Вариант 13: Калькулятор. Формула должна поступать в виде в виде текста (пример: «4/2»); отдельными значениями данных.

Реализация в протоколе передачи данных HTTP с запросами GET- и POST-.

# Поэтапное выполнение практической работы

1. порт, на котором запускается сервер, 8080;
2. количество одновременно обрабатываемых клиентских запросов не ограничено (создание многопоточности);
3. сервер должен распознавать метод запроса и реагировать только на метод GET. Реакция на все остальные методы (POST, PUT, DELETE и др.) не оговаривается и может быть реализована по желанию, но при этом, если будет реализован только один метод GET, на другие методы ваш сервер не должен срабатывать;
4. в заголовке выдаваемого ответа нужно указать корректный тип (text/html) и длину тела сформированного сообщения;
5. в ответ на любой запрошенный ресурс сервер должен выдавать HTMLстраницу с возможностью отображения русских слов.
6. написать процедуру распознавания окончания запроса (наличия пустой строки, в которой есть символ возврата каретки, но нет ни одного значащего текстового символа);
7. по окончании запроса провести анализ полученного метода. Если это GET — перейти к формированию корректного ответа клиенту;
8. сформировать ответ и выдать его клиенту. Корректно завершить соединение с клиентом. При следующем запросе клиент установит новое соединение;
9. должен быть предусмотрен механизм обработки кодировки входящего/исходящего потока данных.

# Код программы

Server.java

package com.company;  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Server {  
 private static ServerSocket *serverSocket*;  
  
 public static void main(String[] args) {  
 try {  
 try {  
 *serverSocket* = new ServerSocket(8080);  
 System.*out*.println("Сервер: Запуск");  
 while (true) {  
 Client client = new Client(*serverSocket*.accept());  
 System.*out*.println("Сервер: Подключение к " + client);  
  
 Thread thread = new Thread(client);  
 thread.start();  
 }  
 } finally {  
 serverSocket.close();  
 }  
 } catch (IOException e) {  
 System.err.println(e);  
 }  
 System.out.println("Сервер: Завершение");  
 }  
}

Client.java

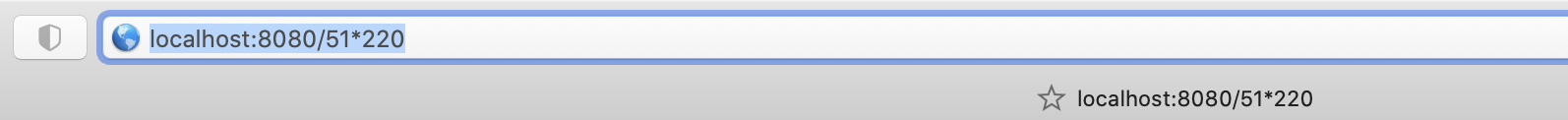
package com.company;  
import java.io.\*;  
import java.net.Socket;  
import java.util.Date;  
import java.util.regex.Pattern;  
  
public class Client implements Runnable {  
  
 private Socket socket;  
 private BufferedReader inStream;  
 private BufferedWriter outStream;  
 private PrintWriter outHeader;  
  
 public Client(Socket clientSocket) {  
 this.socket = clientSocket;  
 }//новая строка  
 //бывшее main  
 public void run() {  
 try {  
 try {  
 //Инициализация  
 inStream = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 outStream = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
 outHeader = new PrintWriter(socket.getOutputStream());  
  
 //Получение запроса  
 String request = inStream.readLine().split(" ")[1].substring(1);  
 System.*out*.println(this + ": Получен запрос: " + request);  
  
 //Заголовки ответа  
 outHeader.println("HTTP/1.1 200 OK");  
 outHeader.println("Server: Java HTTP Server : 1.0");  
 outHeader.println("Date: " + new Date());  
 outHeader.println("Content-type: text/html; charset=utf-8");  
 outHeader.println("Accept-Language: \*");  
 outHeader.println();  
 outHeader.flush();  
  
 System.*out*.println(this + ": Подготовка ответа.");  
  
 //Отображение базового текста  
 outStream.write("<h1>Работу выполнил: Копотов Михаил Алексеевич. ИКБО-12-18</h1>" +  
 "<h2>Номер индивидуального задания: 13</h2>" +  
 "<h2>Текст индивидуального задания: Калькулятор. Формула должна поступать в виде в виде текста (пример: «4/2»)</h2>");  
  
 //Отображение ответа  
 try {  
 var answer = *calculate*(request);  
 outStream.write("<h3 style='color: blue'>Ответ: " + answer + "</h3>");  
 }catch (Exception e){  
 outStream.write("<h3 style='color: red'>Ошибка: " + e + "</h3>");  
 }  
  
 outStream.flush();  
  
 System.*out*.println(this + ": Ответ отправлен.");  
  
 } finally {  
 System.*out*.println(this + ": Остановка");  
 socket.close();  
 inStream.close();  
 outStream.close();  
 }  
 } catch (IOException e) {  
 System.*err*.println(e);  
 }  
 }  
  
 static float calculate(String task) throws Exception {  
 char sign;  
 String aString = "", bString = "";  
  
 int i = 0;  
 while (Character.*isDigit*(task.charAt(i)))  
 aString += task.charAt(i++);  
  
 sign = task.charAt(i++);  
  
 while (i < task.length())  
 bString += task.charAt(i++);  
  
 float a = Float.*parseFloat*(aString);  
 float b = Float.*parseFloat*(bString);  
  
 switch (sign) {  
 case '+':  
 return (a + b);  
 case '-':  
 return (a - b);  
 case '\*':  
 return (a \* b);  
 case '/':  
 return (a / b);  
 default:  
 throw new Exception();  
 }  
 }  
}

# 5 Результат работы

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 1.Информационная страница по адресу localhost:8080/51\*220**

****

**Рисунок 2. Данные, отправленные на сервер**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 3. Результат выполнения операции, заданной с ошибкой.**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 4. Консоль сервера, один раз получившего ошибку**

# 6 Вывод

В данной практической работе мы ознакомились с протоколом передачи файлов прикладного уровня HTTP, а также научились работать с GET- и POST-запросами.

# Список использованных источников

1. https://tproger.ru/translations/java-intro-for-beginners/
2. https://www.sqlitetutorial.net/
3. https://www.jetbrains.com/ru-ru/idea/
4. https://developer.mozilla.org/ru/docs/Web/HTTP/Overview