|  |
| --- |
| 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-01-18 | Маркин В.В. |
| Принял ассистент кафедры ИППО | Строганкова Н.В. |

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

**Содержание**

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

[2. Задание 3](#_Toc528748842)

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

[5. Текст разметки 4](#_Toc528748844)

[6. Результат работы 8](#_Toc528748845)

[7. Вывод 9](#_Toc528748846)

[Список использованных источников 10](#_Toc528748847)

# 1 Цель практической работы

Целью данной работы является знакомство с протоколом передачи файлов прикладного уровня HTTP, а также научиться работать с GET- и POST-запросами.

# 2 Задание

Создать клиент-серверное приложение и с помощью GET-запроса выполнить передачу данных в адресную строку сайта. В качестве входных данных использовать данные из практической работы №6 (поиск максимального числа в последовательности входных данных).

# 3 Поэтапное выполнение практической работы

Технические требования к разрабатываемому серверу следующие:

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 HTML-страницу с возможностью отображения русских слов.

# 4 Текст разметки

Папка logic – расположение всей логики проекта

|  |
| --- |
| Result.java |
| package logic;  public interface Result {  double getResult();  } |

|  |
| --- |
| Math.java |
| package logic;  import java.util.ArrayList;  import java.util.List;  public class Math implements Result{  Double max;  String text;  double[] nums;  List<Double> lineOfNums;  private int i;  /\*  Создание метода для возвращения результата.  Входных параметров: нет  Выходные данные: результат вычислений.  \*/  @Override  public double getResult() {  //Вернуть параметр - результат вычислений.  return max;  }  //Конструктор для строки  public Math(String text) {  this.text = text;  System.out.println("Конструктор для строки:"+this.text);  }  public void Parser(){  ArrayList<Double>list = new ArrayList<>();  //Массив чисел  String[] temp = text.split(",");  int length = temp.length;  //Парсим числа в целочисленные и добавление в лист  for(int j = 0; j < temp.length; j++){  // i = Integer.parseInt(temp[j]);  double k = Double.parseDouble(temp[j]);  list.add(k);  }  //Отправляем в метод сортировки  //Без ожидания  maxNumOfList(list);  }  public void maxNumOfList(ArrayList<Double>list){  max = list.get(0);  for (int i = 1; i < list.size(); i++){  if(list.get(i) > max){  max = list.get(i);  }  }  }  } |

Серверная часть

|  |
| --- |
| SimpleHttpServer.java |
| package server;  import java.io.IOException;  import java.net.ServerSocket;  public class SimpleHttpServer {  /\*\*  \* @param args  \*/  public static void main(String[] args) {  try {  ServerSocket serverConnect = new ServerSocket(8080);  System.out.println("Сокет создан на порту 8080 - ждем запросов от клиентов");  // начинаем слушать запросы  while (true) {  ClientHandler myServer = new ClientHandler(serverConnect.accept());  System.out.println("Соединение установлено");  // создаем отдельный поток для обработки запроса и формирования ответа  Thread thread = new Thread(myServer);  thread.start();  }  } catch (IOException e) {  e.printStackTrace();  }  }  } |

|  |
| --- |
| ClientHandler.java |
| package server;  import logic.Math;  import java.io.\*;  import java.net.Socket;  import java.util.Date;  import java.util.StringTokenizer;  public class ClientHandler implements Runnable {  private static Socket clientDialog;  public ClientHandler(Socket client) {  this.clientDialog = client;  }  //Создание ссылки math объетка logic.Math  //При работе в методе run() будет происходить  //инъекция данных в данный класс для вычисления результата  Math math;  @Override  public void run() {  BufferedReader in = null;  PrintWriter out = null;  BufferedOutputStream dataOut = null;  String fileRequested = null;  try {  // канал чтения из сокета  in = new BufferedReader(new InputStreamReader(clientDialog.getInputStream()));  // канал записи в сокет (для HEADER)  out = new PrintWriter(clientDialog.getOutputStream());  // канал записи в сокет (для данных)  dataOut = new BufferedOutputStream(clientDialog.getOutputStream());  // первая строка запроса  String input = in.readLine();  // разбираем запрос по токенам  StringTokenizer parse = new StringTokenizer(input);  String method = parse.nextToken().toUpperCase(); // получаем HTTP метод от клиента  // текст запроса от клиента  fileRequested = parse.nextToken().toLowerCase();  System.out.println("Method: " + method);  System.out.println("Request: " + fileRequested.substring(1));  // В данной точке кода добавить вхождение в файл и вычислить ответ.  math = new Math(fileRequested.substring(1));  math.Parser();  // Тестовые данные. Вывод данных на монитор для проверки  System.out.println("Данные с парсера:"+math.getResult());  // пока поддерживаем GET and HEAD запросы  if (method.equals("GET") || method.equals("HEAD")) {  String content = getContentType(fileRequested);  //Записать в переменную body полученное значение, а не значение, полученное в ходе запроса  String numbers = "Input numbers: " + getBody(fileRequested.substring(1));  String body = "Result of counting: " + getBodyResult(Double.toString(math.getResult()));  String param = "The work was performed by: Markin Viktor Vladimirovich <br> Group Number: IKBO-01-18\n" +  "Individual task number: 1<br>\n" +  "Text of the individual task: \"Search for the maximum number\"<br>";  //String body = Double.toString(math.getResult());  if (method.equals("GET")) {  // GET method - возвращаем ответ  // шлем HTTP Headers  out.println("HTTP/1.1 200 OK");  out.println("Server: Java HTTP Server : 1.0");  out.println("Date: " + new Date());  out.println("Content-type: " + content);  //Длина ответа - эхо запроса без первого "/"  out.println("Content-length: " + body.length() + numbers.length() + param.length());  out.println(); // Пустая строка между headers и содержимым!  out.flush();  dataOut.write(param.getBytes(), 0, param.length());  dataOut.write(numbers.getBytes(), 0, numbers.length());  dataOut.write(body.getBytes(), 0, body.length());  //Отправляем тестовые данные  // dataOut.write();  dataOut.flush();  }  System.out.println("Ответ отослан: " + body);  }  } catch (IOException e) {  e.printStackTrace();  }  }  // Возвращаем поддерживываемый MIME Types  private String getContentType(String fileRequested) {  // if (fileRequested.endsWith(".htm") || fileRequested.endsWith(".html"))  return "text/html";  // else  // return "text/plain";  }  private String getBody(String request) {  return "<b>" + request + "</b>" + "<br>";  }  private String getBodyResult(String result){  return "<b>" + result + "</b>";  }  } |

# 5 Результат работы

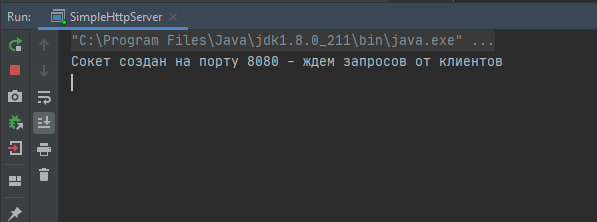


Рис. 1. Отзыв приложения в терминале

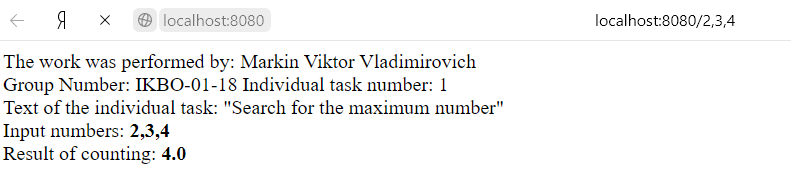


Рис. 2. Результат выполнения работы на html странице

# 6 Вывод

В процессе выполнения данной работы, мы получили практические навыки работы со средой разработки IntellijIDEA. Также научились создавать простейшее серверное приложения на языке программирования Java.

Также познакомились с протоколами передачи файлов прикладного уровня HTTP, а также научились работать с GET- и POST- запросами.

# Список использованных источников

1. <https://habr.com/ru/post/330676/>
2. <http://mindhalls.ru/client-server-application-in-java/>
3. <https://hr-vector.com/java/klient-server>
4. <https://javarush.ru/groups/posts/2529-chastjh-5-servletih-pishem-prostoe-veb-prilozhenie>