МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет информационных технологий и робототехники***

Кафедра программного обеспечения информационных технологий

**Отчет по лабораторной работе № 6**

по дисциплине: ”Компьютерные системы и сети”

на тему: **”ПРОТОКОЛ HTTP*”***

Выполнили**:** Студент группы 10701322 Сахаревич И.Н.

Студент группы 10701322 Бородкин Д.В.

Приняла**:** Белова С.В.

Минск 2024

**Лабораторная работа №6**

**Цель работы:**

Изучить работу протокола HTTP. Разработать web-сервер и web-клиент.

**Задание 1.** **HTTP-сервер**

Написать консольное приложение, реализующее основные функции HTTP-сервера.

Должен поддерживаться запрос GET. Сервер должен быть сконфигурирован на

определенный каталог, где расположены html-файлы. На экране отображать весь протокол

общения HTTP клиента с HTTP сервером: все приходящие от клиента запросы и все

отправляемые ему ответы.

Тестирование и демонстрацию программы-сервера производить при помощи Web-

браузера и своего Web-клиента.

**Код:**

using System;

using System.IO;

using System.Net;

using System.Text;

class SimpleHttpServer

{

static void Main()

{

string rootDirectory = @"E:\BNTU\2курс\КСИС\Lab6\serverRoot";

HttpListener listener = new HttpListener();

listener.Prefixes.Add("http://localhost:8080/");

Console.WriteLine("Server started. Listening for connections...");

listener.Start();

while (true)

{

HttpListenerContext context = listener.GetContext();

HttpListenerRequest request = context.Request;

HttpListenerResponse response = context.Response;

Console.WriteLine("Client connected. Request URL: " + request.Url);

string filePath = rootDirectory + request.Url.LocalPath;

if (File.Exists(filePath))

{

byte[] content = File.ReadAllBytes(filePath);

response.ContentLength64 = content.Length;

response.OutputStream.Write(content, 0, content.Length);

response.OutputStream.Close();

Console.WriteLine("File sent to client.");

}

else

{

response.StatusCode = (int)HttpStatusCode.NotFound;

response.Close();

Console.WriteLine("File not found.");

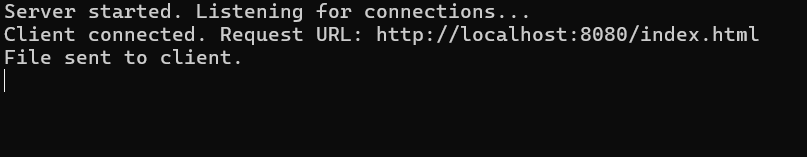
}

}

}

}

**Результаты:**

****

**Задание 2.** **HTTP-клиент**

Написать консольное приложение, реализующее функции простого HTTP-клиента.

Получение Web-страницы от HTTP-сервера должно осуществляться по заданному URI-

адресу. В HTTP-запрос добавлять заголовок User-Agent и другие заголовки. Должен

поддерживаться запрос GET, а также наиболее распространенные коды ответов.

Отображать на экране весь протокол общения HTTP клиента с HTTP сервером.

Полученные от сервера данные выводить в виде HTML-кода на экран или в файл.

Отображение полученных данных в форматированном виде не обязательно.

Тестирование и демонстрацию HTTP клиента производить при помощи запроса к

реальному web-серверу, расположенному в Internet, установленному в локальной сети или при помощи запроса к web-серверу, написанному в предыдущем задании.

**Код:**

using System;

using System.IO;

using System.Net.Sockets;

using System.Text;

class SimpleHttpClient

{

static void Main()

{

//string url = "http://example.com/index.html"; // Заданный URI-адрес

//string host = "example.com";

//string html = "index.html";

//int port = 80;

string url = "http://localhost:8080/index.html";

string host = "localhost";

string html = "index.html";

int port = 8080;

TcpClient client = new TcpClient();

client.Connect(host, port);

NetworkStream stream = client.GetStream();

string request = "GET /" + html + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"User-Agent: MySimpleHttpClient\r\n" +

"\r\n";

byte[] requestData = Encoding.UTF8.GetBytes(request);

stream.Write(requestData, 0, requestData.Length);

byte[] responseData = new byte[4096];

int bytesRead = stream.Read(responseData, 0, responseData.Length);

string response = Encoding.UTF8.GetString(responseData, 0, bytesRead);

Console.WriteLine(response);

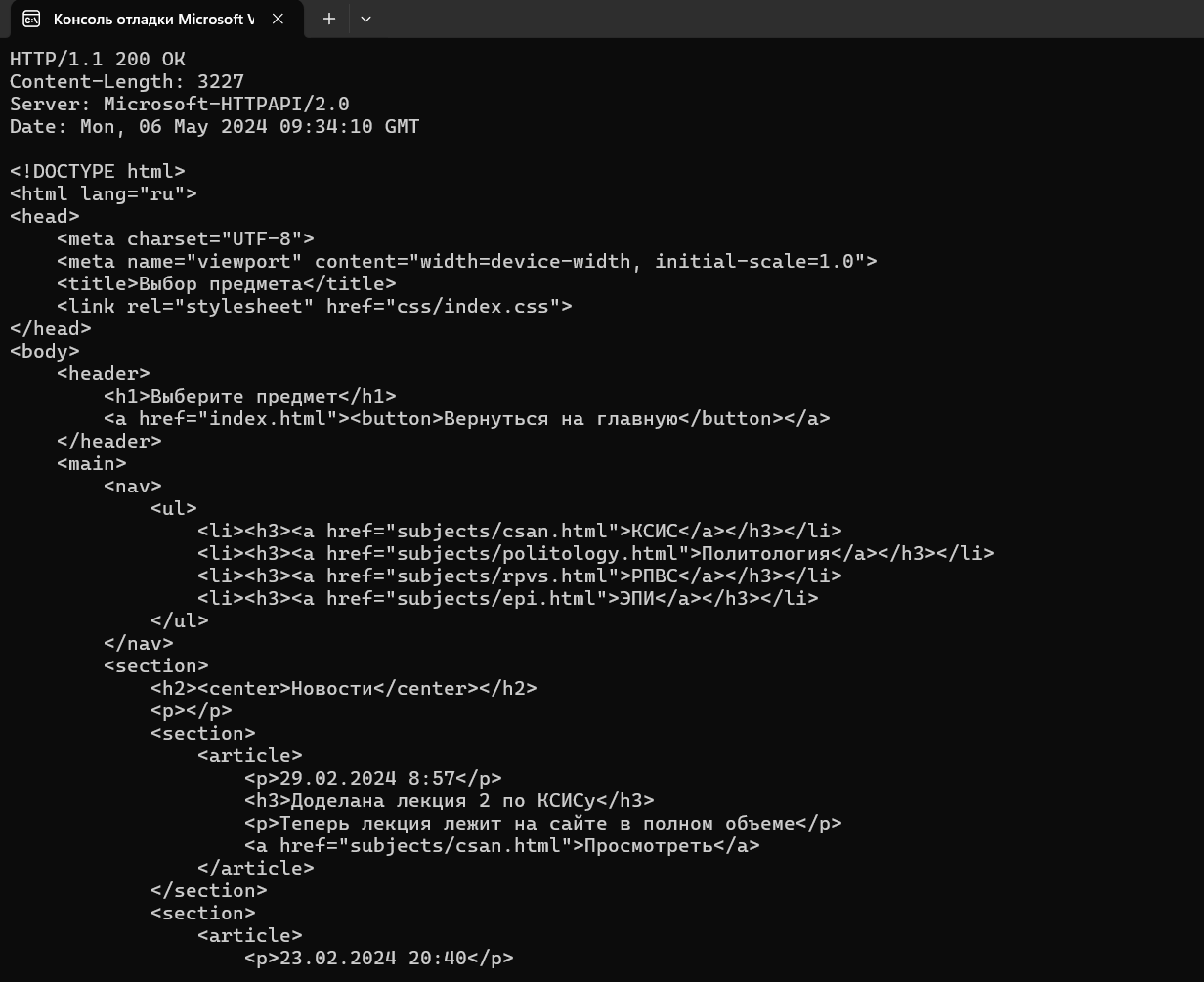
File.WriteAllText("response.html", response);

client.Close();

}

}

**Результаты:**

****

**Контрольные вопросы:**

**На каком уровне стека протоколов работает HTTP?**

В стеке протоколов HTTP находится на прикладном уровне.

**Для чего предназначен протокол HTTP?**

HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы.

**Какой номер порта закреплен за HTTP по умолчанию?**

80

**Какова структура запроса HTTP-клиента? Что такое метод? Перечислите основные методы.**

HTTP-запрос состоит из трех основных частей:

1. **Стартовая строка (Start Line)**: Она включает в себя метод запроса, URL и версию протокола HTTP.
2. **Заголовки (Headers)**: Они содержат дополнительную информацию о запросе или ответе, такую как тип содержимого или информацию об авторизации.
3. **Тело сообщения (Message Body)**: Это необязательная часть запроса, которая содержит данные, отправляемые на сервер. В запросах GET тела сообщения обычно нет.

**Метод HTTP** определяет действие, которое должно быть выполнено на указанный ресурс. Некоторые из основных методов HTTP включают:

* **GET**: Запрашивает данные от указанного ресурса.
* **POST**: Отправляет данные для хранения на указанный ресурс.
* **PUT**: Обновляет указанный ресурс данными из запроса.
* **DELETE**: Удаляет указанный ресурс.
* **HEAD**: Аналогичен GET, но сервер возвращает только заголовки ответа, а не тело ответа.
* **OPTIONS**: Возвращает поддерживаемые методы

**Какова структура ответа HTTP-сервера? Что такое код состояния? Перечислите наиболее распространенные коды состояний?**

HTTP-ответ также состоит из трех основных частей, подобно HTTP-запросу:

1. **Стартовая строка (Start Line)**: Она включает в себя версию протокола HTTP, код состояния и текстовое описание состояния.
2. **Заголовки (Headers)**: Они содержат дополнительную информацию о запросе или ответе, такую как тип содержимого или дата.
3. **Тело сообщения (Message Body)**: Это необязательная часть ответа, которая содержит данные, отправляемые клиенту.

**Код состояния HTTP** - это трехзначное число в стартовой строке ответа, которое указывает на результат выполнения запроса. Некоторые из наиболее распространенных кодов состояния HTTP включают:

* **200 OK**: Запрос успешно выполнен, а ответное сообщение содержит ожидаемые данные.
* **301 Moved Permanently**: Запрашиваемый ресурс был окончательно перемещен на новый URI, указанный в поле Location заголовка.
* **400 Bad Request**: Сервер не понимает запрос из-за неверного синтаксиса.
* **401 Unauthorized**: Для доступа к ресурсу требуется аутентификация.
* **403 Forbidden**: Сервер понимает запрос, но он отказывается его выполнять.
* **404 Not Found**: Сервер не может найти запрашиваемый ресурс.
* **500 Internal Server Error**: На сервере произошла неожиданная ошибка.

**Какие заголовочные поля могут содержаться в HTTP-запросе и HTTP-ответе?**

**HTTP-запрос:**

* **Accept**: Типы контента, которые клиент может обрабатывать.
* **Accept-Language**: Предпочитаемые языки клиента для ответа.
* **Authorization**: Информация об авторизации для доступа к ресурсам.
* **Host**: Доменное имя и порт сервера.
* **User-Agent**: Информация о клиенте, инициировавшем запрос.
* **Content-Type**: MIME-тип тела запроса.
* **Content-Length**: Размер тела запроса в байтах.

**HTTP-ответ:**

* **Content-Type**: MIME-тип тела ответа.
* **Content-Length**: Размер тела ответа в байтах.
* **Date**: Дата и время, когда сообщение было отправлено.
* **Server**: Имя и версия веб-сервера.
* **Set-Cookie**: Куки, которые клиент должен сохранить.
* **WWW-Authenticate**: Информация об авторизации для доступа к ресурсам.

**Какие классы .NET можно использовать для работы с протоколом HTTP?**

В .NET Framework существует несколько классов, которые можно использовать для работы с протоколом HTTP:

* **HttpClient**: Этот класс предоставляет методы для отправки HTTP-запросов и получения HTTP-ответов от ресурса, указанного URI. Он используется для отправки запросов GET, POST, PUT, DELETE и других.
* **HttpWebRequest и HttpWebResponse**: Эти классы предоставляют более гибкий и низкоуровневый доступ к HTTP-запросам и ответам. Они позволяют управлять такими вещами, как cookies, протоколы аутентификации и прокси-серверы.
* **WebClient**: Это более высокоуровневый класс, который обеспечивает общие методы для отправки данных на и получения данных из любого ресурса, указанного URI.
* **HttpRequestMessage и HttpResponseMessage**: Эти классы представляют HTTP-запросы и HTTP-ответы, включая заголовки и данные.
* **HttpListener**: Этот класс слушает HTTP-запросы от клиентов и предоставляет ответы на них.

**Какие ресурсы можно адресовать с помощью URI? Приведите примеры широко распространенных схем, используемых в URI.**

* **http**: Используется для обращения к веб-страницам. Например, http://example.com.
* **https**: Это зашифрованная версия протокола HTTP. Например, https://example.com.
* **ftp**: Используется для доступа к файлам на FTP-серверах. Например, ftp://example.com.
* **mailto**: Используется для ссылок на адреса электронной почты. Например, mailto:example@example.com.
* **file**: Используется для доступа к локальным файлам на компьютере. Например, file:///C:/example.txt.

**Какова структура URI?**

Схема, домен, путь, запрос и фрагмент

**Как осуществляется работа с URI адресами в .NET?**

Через System.Uri (часть System.Net)

**Выводы:**

В ходе выполнения этой лабораторной работы, мы изучили работу HTTP-протокола и научились разрабатывать собственные web-клиенты и web-сервера.