**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №8**

**Дисциплина: Сети ЭВМ и телекоммуникации**

**Тема: Программирование протокола HTTP**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: ст. пр. ПО и ВТАС

Федотов Евгений Александрович

**Белгород 2020**

**Цель работы:** изучить протокол HTTP и составить программу согласно заданию.

**Краткие теоретические сведения**

**HTTP** (Hyper Text Transfer Protocol – протокол передачи гипертекста) – протокол прикладного уровня стека протоколов TCP/IP, предназначенный для передачи данных по сети с использованием транспортного протокола TCP. Текущая версия протокола HTTP v1.1, его спецификация приводится в документе RFC 2616.

Протокол HTTP может использоваться также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP или XML-RPC.

Основой HTTP является технология «клиент-сервер». HTTP клиенты отсылают HTTP-запросы, которые содержат метод, обозначающий потребность клиента. Также такие запросы содержат универсальный идентификатор ресурса, указывающий на желаемый ресурс. Обычно такими ресурсами являются хранящиеся на сервере файлы. По умолчанию HTTP-запросы передаются на порт 80. HTTP сервер отсылает коды состояния, сообщая, успешно ли выполнен HTTP-запрос или же нет.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т.д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

Унифицированный идентификатор ресурса представляет собой сочетание унифицированного указателя ресурса (Uniform Resource Locator, URL) и унифицированного имени ресурса (Uniform

Resource Name, URN). Например:

URI= http://iitus.bstu.ru/to\_schoolleaver/230400

URL= http://iitus.bstu.ru/

URN= to\_schoolleaver/230400

Метод протокола HTTP – это команда, передаваемая HTTP клиентом HTTP-серверу. В табл. 8.1. перечислены некоторые методы, определенные в протоколе HTTP v1.1. Полный список методов HTTP

v1.1. содержится в документе RFC 2616.

**Основные методы HTTP v1.1**

|  |  |
| --- | --- |
| Метод | Назначение |
| GET | Используется для запроса содержимого ресурса, на который указывает URI, содержащийся в запросе |
| HEAD | Используется для извлечения метаданных или проверки наличия ресурса, на который указывает URI, содержащийся в запросе |
| POST | Применяется для передачи данных заданному ресурсу. Данный метод предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого |
| PUT | Применяется для передачи данных заданному ресурсу. Данный метод предполагает, что передаваемое клиентом содержимое соответствует находящемуся по данному URI ресурсу |
| OPTIONS | Используется для определения возможностей HTTP сервера или параметров соединения для конкретного ресурса |
| DELETE | Применяется для удаления ресурса, на который указывает URI |

Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Название метода чувствительно к регистру. Каждый сервер обязан поддерживать как минимум методы GET и HEAD. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (NotImplemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (MethodNotAllowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Код состояния HTTP представляет собой целое число из трех цифр. Первая цифра указывает на класс состояния:

• информационные сообщения;

• успешное выполнение;

• переадресация;

• ошибка клиента;

• ошибка сервера.

Полный список статусов HTTP v1.1. содержится в документе RFC 2616. Примеры:

201 Webpage Created

403 Access allowed only for registered users

Каждое HTTP-сообщение состоит из трех частей, которые передаются в следующем порядке:

1. Стартовая строка – определяет тип сообщения;

2. Заголовки – характеризуют тело сообщения, параметры

передачи и прочие сведения;

3. Тело сообщения – непосредственно данные сообщения.

Стартовые строки HTTP-сообщения различаются для запроса и ответа. Стартовая строка HTTP-запроса имеет следующий формат:

Метод URI HTTP/Версия, где метод - название запроса,

URI определяет путь к запрашиваемому документу, версия - пара разделённых точкой арабских цифр.

Стартовая строка HTTP-ответа имеет следующий формат:

HTTP/Версия КодСостояния Пояснение.

Заголовок HTTP представляет собой строку в HTTP-сообщении, содержащую разделенную двоеточием пару параметр-значение. Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой. Все заголовки разделяются на четыре основных группы:

1. Основные заголовки (General Headers) – должны включаться в любое сообщение клиента и сервера.

2. Заголовки запроса (Request Headers) – используются только в запросах клиента.

3. Заголовки ответа (Response Headers) – только для ответов от сервера.

4. Заголовки сущности (Entity Headers) – сопровождают каждую сущность сообщения.

Тело HTTP-сообщения, если оно присутствует, используется для передачи данных, связанных с запросом или ответом. Чтобы понять, как аботает протокол HTTP, рассмотрим пример полученной HTML-страницы с HTTP-сервера.

HTTP-запрос:

GET /to\_schoolleaver/230400 HTTP/1.1

Host: iitus.bstu.ru

User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;

Trident/5.0)

Accept: text/html

Accept-Language: ru

Connection: close

HTTP-ответ:

HTTP/1.1 200 OK

Date: Fri, 16 Dec 2012 13:45:00 GMT

Server: Apache

Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

(далее следует запрошенная HTML-страница)

Протокол HTML позволяет достаточно легко создавать клиентские приложения. Возможности протокола можно расширить благодаря внедрению своих собственных заголовков, с помощью которых можно получить необходимую функциональность при решении нетривиальной задачи. При этом сохраняется совместимость с другими клиентами и серверами: они будут просто игнорировать неизвестные им заголовки.

Протокол HTTP устанавливает отдельную TCP-сессию на каждый запрос; в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются cookies; причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP или по файловым протоколам тип файла (точнее, тип содержащихся в нём данных) определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «Content-Type: тип/подтип», позволяющую клиенту однозначно определить, каким образом обрабатывать присланные данные. Это особенно важно при работе с CGI-скриптами, когда расширение имени файла указывает не на тип присылаемых клиенту данных, а на необходимость запуска данного файла на сервере и отправки клиенту результатов работы программы, записанной в этом файле (при этом один и тот же файл в зависимости от аргументов запроса и своих собственных соображений может порождать ответы разных типов — в простейшем случае картинки в разных форматах).

**Задание к работе**

1. Разработать программу, позволяющую принимать запрос на выдачу страницы от интернета браузера и формировать ответ в зависимости от запроса. Реализовать методы GET и HEAD на стороне сервера.
2. Передать браузеру в сообщении тестовую страницу (определяется программистом) запрошенную страницу, или код ошибки если страница не найдена.

Программы должны быть написаны на языке программирования C.

**Результат работы программы**

|  |  |
| --- | --- |
|  |  |

**Изображение выглядит как снимок экрана

Автоматически созданное описание**

**Вывод:**

В данной работе был реализован простейший HTTP сервер, написанный с использованием библиотек Winsock. Эта библиотека позволяет быстро и удобно писать сетевые приложения под OS Windows.

*Приложение*

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

…

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp29{

public partial class MainWindow : Window{

public MainWindow(){

InitializeComponent();

}

HttpListener Server = new HttpListener();

void ServerCycle(){

while (true){

HttpListenerContext Context = Server.GetContext();

HttpListenerResponse Response = Context.Response;

switch (Context.Request.HttpMethod)

{

case "GET":

{

string answer = "Hello, Student!";

Response.ContentLength64 = Encoding.UTF8.GetByteCount(answer);

Response.OutputStream.Write(Encoding.UTF8.GetBytes(answer), 0, Encoding.UTF8.GetByteCount(answer));

break;

}

case "HEAD":

{

string answer = "Hello, Student, it's header!";

Response.AddHeader("Data", answer);

break;

}

default:

break;

}

Context.Response.Close();

}

}

private void button\_Click(object sender, RoutedEventArgs e){

Server.Prefixes.Add("http://127.0.0.1:7767/");

Server.Start();

new Thread(ServerCycle).Start();

button.Visibility = Visibility.Collapsed;

}

}

}