Белгородский государственный технологический университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №8 по теме: «Программирование протокола HTTP»

**Выполнил:**  
студент группы ПВ-31  
Адаменко И. И.

**Проверил:**старший преподаватель  
Федотов Е. А.

Белгород  
2015

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

**Задание:**

1. Разработать программу, позволяющую принимать запрос на выдачу страницы от интернет-браузера и формировать ответ в зависимости от запроса. Реализовать методы GET и HEAD на стороне сервера.
2. Передать браузеру в сообщении тестовую страницу или код ошибки, если таковая не найдена.
3. Программы должны быть написаны на языке программирования C#.

# Теоретическая часть

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.

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 таблице ниже перечислены некоторые методы, определенные в протоколе HTTP v1.1. Полный список методов HTTP v1.1. содержится в документе RFC 2616.

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). Еслисерверу 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.

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.

Тело HTTP-сообщения, если оно присутствует, используется для передачи данных, связанных с запросом или ответом.

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.

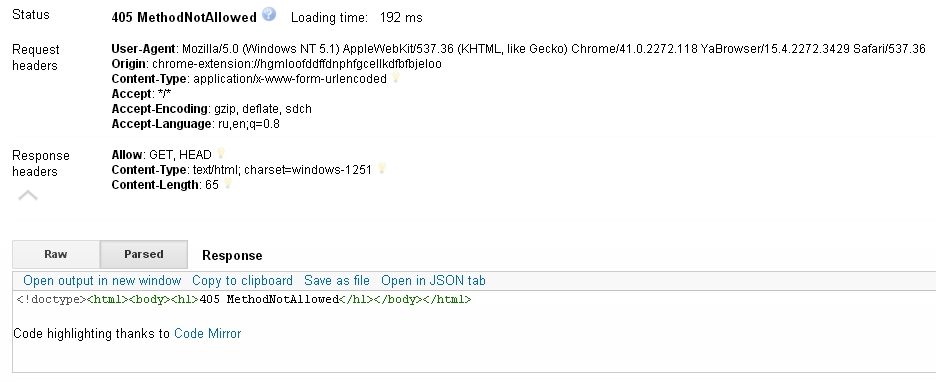
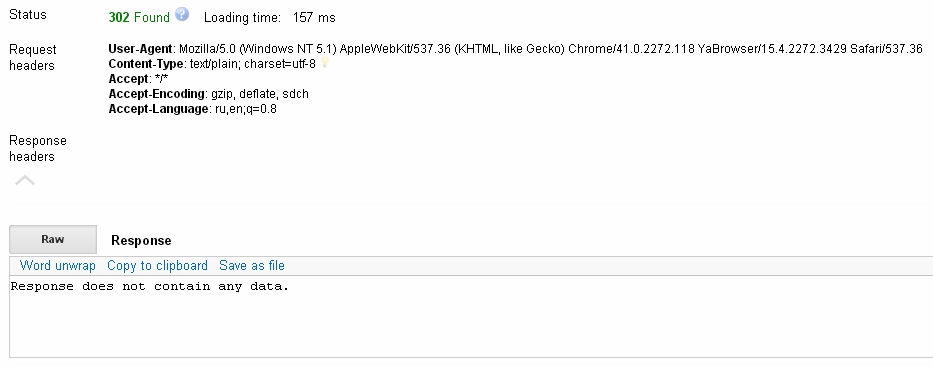
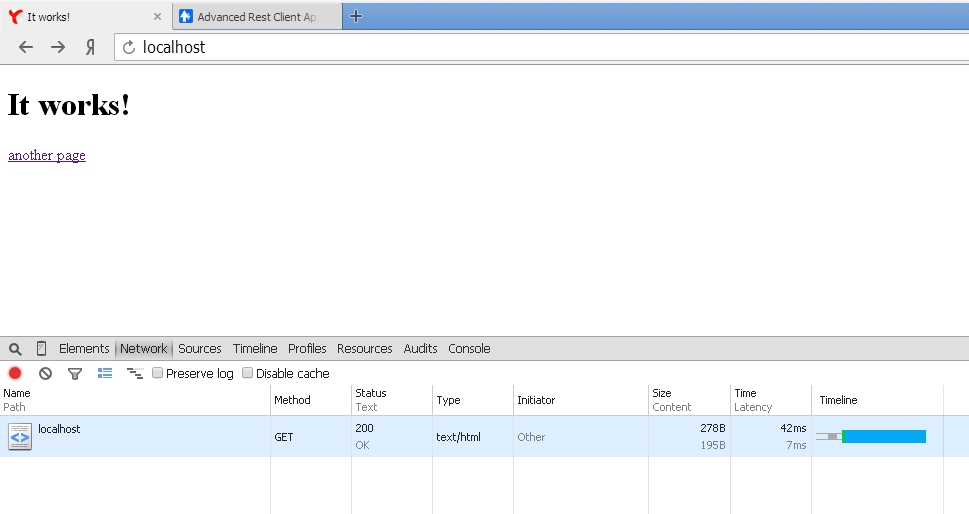
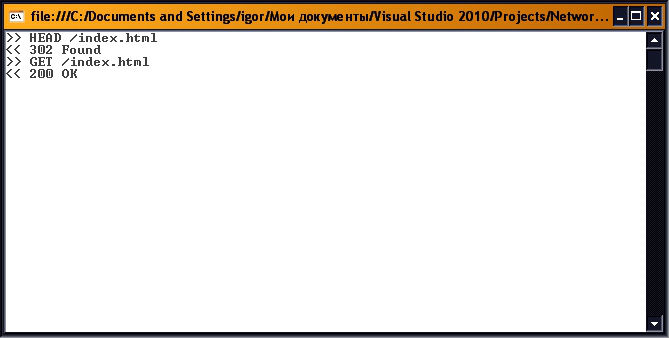
Протокол HTTP устанавливает отдельную TCP-сессию на каждый запрос; в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. 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).

Кроме того, HTTP позволяет клиенту прислать на сервер параметры, которые будут переданы запускаемому CGI-скрипту. Для этого же в HTML были введены формы.

Перечисленные особенности HTTP позволили создавать поисковые машины (первой из которых стала AltaVista, созданная фирмой DEC), форумы и Internet-магазины. Это коммерциализировало Интернет, появились компании, основным полем деятельности которых стало предоставление доступа в Интернет (провайдеры) и создание сайтов.

# Скриншоты работы приложения



# Код программы

Основной файл:

1. **namespace** Lab8
2. {
3. **class** Program
4. {
5. **static** **void** Main(**string**[] args)
6. {
7. **new** Server(80);
8. }
9. }
10. }

Файл с классом, отвечающим за «прослушивание» выбранного порта:

1. **using** System.Net.Sockets;
2. **using** System.Net;
4. **namespace** Lab8
5. {
6. **class** Listener
7. {
8. TcpListener listener;
10. **public** Listener(**int** Port)
11. {
12. listener = **new** TcpListener(IPAddress.Any, Port);
13. listener.Start();
15. **while** (**true**)
16. {
17. var newClient = listener.AcceptTcpClient();
18. **new** Server(newClient);
19. }
20. }
21. }
22. }

Файл с классом сервера:

1. **using** System;
2. **using** System.Text;
3. **using** System.Net.Sockets;
4. **using** System.Net;
5. **using** System.IO;
7. **namespace** Lab8
8. {
9. **class** Server
10. {
11. **string** contenttype = "Content-Type: text/html; charset=windows-1251\n";
13. **public** Server(TcpClient client)
14. {
15. var buffer = **new** **byte**[1024];
16. var sizeData = client.GetStream().Read(buffer, 0, buffer.Length);
17. var request = Encoding.ASCII.GetString(buffer, 0, sizeData);
18. var line = request.Split('\n')[0].Split(' ');
20. **if** (line.Length < 2)
21. {
22. SendError(client, 400, "GET");
23. **return**;
24. }
26. var method = line[0];
27. var uri = line[1];
29. **if** (uri.EndsWith("/"))
30. {
31. uri += "index.html";
32. }
34. **if** (method != "GET" && method != "HEAD")
35. {
36. SendError(client, 405, "GET");
37. **return**;
38. }
39. **else**
40. {
41. ClientLog(method + " " + uri);
43. var path = "www/" + uri;
44. **if** (!File.Exists(path))
45. {
46. SendError(client, 404, method);
47. **return**;
48. }
50. FileStream fs;
51. **try**
52. {
53. fs = **new** FileStream(path, FileMode.Open,  
     FileAccess.Read, FileShare.Read);
54. }
55. **catch** (Exception)
56. {
57. SendError(client, 500, method);
58. **return**;
59. }
61. **if** (method == "GET")
62. {
63. var headers = "HTTP/1.1 200 OK\n" + contenttype +  
     "Content-Length: " + fs.Length + "\n\n";
65. ServerLog("200 OK");
66. SendResponse(client, headers);
68. **while** (fs.Position < fs.Length)
69. {
70. sizeData = fs.Read(buffer, 0, buffer.Length);
71. client.GetStream().Write(buffer, 0, sizeData);
72. }
73. }
74. **else**
75. {
76. var headers = "HTTP/1.1 302 Found\n" + contenttype + "\n\n";
77. ServerLog("302 Found");
78. SendResponse(client, headers);
79. }
80. }
82. client.Close();
83. }
85. **private** **void** ClientLog(**string** str)
86. {
87. Console.WriteLine(">> " + str);
88. }
90. **private** **void** ServerLog(**string** str)
91. {
92. Console.WriteLine("<< " + str);
93. }
95. **private** **void** SendResponse(TcpClient client, **string** response)
96. {
97. var buffer = Encoding.ASCII.GetBytes(response);
98. client.GetStream().Write(buffer, 0, buffer.Length);
99. }
101. **private** **void** SendError(TcpClient client, **int** code, **string** method)
102. {
103. var error = code.ToString() + " " + ((HttpStatusCode)code).ToString();
104. **string** body = "", allow = "";
106. **if** (method == "GET")
107. {
108. body = "<!doctype><html><body><h1>" +  
      error + "</h1></body></html>";
109. }
111. **if** (code == 405 || code == 501)
112. {
113. allow = "Allow: GET, HEAD\n";
114. }
116. var response = "HTTP/1.1 " + error +
117. "\n" + allow + contenttype +  
      "Content-Length: " + body.Length.ToString() +
118. "\n\n" + body;
120. ServerLog(error);
121. SendResponse(client, response);
122. client.Close();
123. }
124. }
125. }

# Ответы на вопросы

1. **Как расшифровывается аббревиатура HTTP?**Hyper Text Transfer Protocol — протокол передачи гипертекста.
2. **Какой уровень занимает протокол в стеке TCP/IP?**HTTP — протокол прикладного уровня стека протоколов TCP/IP, предназначенный для передачи данных по сети с использованием транспортного протокола TCP.
3. **На какой технологии построен протокол HTTP?**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.
4. **Какие преимущества протокола HTTP?**

Данный протокол очень прост в реализации, хорошо расширяется благодаря внедрению собственных заголовков, добавляющие функциональности приложению, и которые будут игнорироваться другими приложениями, посчитавшими их неизвестными.

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.

1. **Какие недостатки протокола HTTP?**

Недостатком протокола является избыточность передаваемой информации, и как следствие, большой размер сообщений по сравнению с передачей двоичных данных.

Также к недостаткам стоит отнести отсутствие навигации. У протокола отсутствуют средства навигации среди ресурсов сервера. Клиент не может, как в FTP запросить список доступных файлов. Протокол предполагает, что пользователю уже известен URI интересующего его ресурса. Эта особенность достаточно прозрачна для пользователя, но неудобна для приложения, которому это иногда требуется. Разработчиками это решается вводом дополнительных компонентов. Со стороны клиента это может быть, например, веб-паук, проходящий по всем гиперссылкам документа, и собирающий данную информации. Со стороны сервера это, например, карта сайта — специальная страница с перечислением доступных клиенту ресурсов. Карта сайта может использоваться как пользователем, так и роботами-пауками поисковых систем, уменьшая для них глубину просмотра—минимально необходимое количество переходов с главной страницы.

1. **Какие методы существуют в протоколе HTTP?**

|  |  |
| --- | --- |
| **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. |

1. **Какие нововведения содержит версия HTTP 1.1?**Режим «постоянного соединения»: TCP-соединение может оставаться открытым после отправки ответа на запрос, что позволяет посылать несколько запросов за одно соединение. Клиент теперь обязан посылать информацию об имени хоста, к которому он обращается, что сделало возможной более простую организацию виртуального хостинга.
2. **Какова структура HTTP-сообщения? Охарактеризуйте каждый элемент.**

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.
4. **Какие существуют классы кодов состояний?**

* 1xx Informational («Информационный»);
* 2xx Success («Успех»);
* 3xx Redirection («Перенаправление»);
* 4xx Client Error («Ошибка клиента»);
* 5xx Server Error («Ошибка сервера»).

1. **Какие существуют группы заголовков HTTP?**

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.
5. **Что такое cookie-файлы? Для чего они используются?**

Куки — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

* аутентификации пользователя;
* хранения персональных предпочтений и настроек пользователя;
* отслеживания состояния сеанса доступа пользователя;
* ведения статистики о пользователях.

1. **Что такое HTTP referrer? Для чего он используется?**

Referer — в протоколе HTTP один из заголовков запроса клиента. Содержит URL источника запроса. Если перейти с одной страницы на другую, referer будет содержать адрес первой страницы. Часто на HTTP-сервере устанавливается программное обеспечение, анализирующее referer и извлекающее из него различную информацию. Так, например, владелец веб-сайта получает возможность узнать, по каким поисковым запросам, как часто и на какие именно страницы попадают люди.