Использование сетевых протоколов HTTP, SMTP, FTP

Обзор HTTP протокола

HTTP – протокол прикладного уровня поверх **TCP/IP**, используемый для передачи гипертекста в WWW и локальных сетях. В основном используется для Помимо передачи содержимого веб-страниц, протокол используется приложениями для обмена информацией. В основе протокола лежат запросы, заголовки и коды результатов. В протоколе всегда выражены 2 стороны - сервер и клиент. Клиент передает запрос в виде:

начальная строка
заголовок (или заголовки)
тело сообщения

, сервер возвращает результат в виде:

начальная строка с кодом результата (или ошибки)
заголовок (или заголовки)
тело сообщения

Клиентами в WWW являются браузеры, например IE или Opera. В качестве серверов могут выступать **HTTP-сервера**, например **Apache** и **IIS**.

Начальная строка запроса клиента - это метод, запрашиваемый **URI** и версия протокола(HTTP-Version)**. HTTP–заголовки** можно разделить на 3 группы – заголовки запроса, заголовки ответа и заголовки, которые встречаются и в запросе, и в ответе. Поля заголовка запроса позволяют клиенту передавать серверу дополнительную информацию о запросе и о самом клиенте. Названия заголовков запросов:

	Заголовок	Назначение
1	**Accept**	список поддерживаемых браузером типов содержимого в порядке их предпочтения данным браузером
2	**Accept-Charset**	Поддерживаемая кодировка. Имеет значение для сервера, который может выдавать один и тот же документ в разных кодировках
3	**Accept-Encoding**	Поддерживаемый тип кодирования. Имеет значение для сервера, который может кодировать один и тот же документ по-разному
4	**Accept-Language**	Поддерживаемый язык. Имеет значение для сервера, который может выдавать один и тот же документ в разных языковых версиях
5	**Authorization**	
6	**From**	
7	**Host**	имя хоста, с которого запрашивается ресурс
8	**If-Modified-Since**	
9	**If-Match**	
10	**If-None-Match**	
11	**If-Range**	
12	**If-Unmodified-Since**	
13	**Max-Forwards**	
14	**Proxy-Authorization**	
15	**Range**	
16	**Referer**	URL, с которого перешли на этот ресурс
17	**User-Agent**	браузер

Начальная строка ответа сервера - это строка состояния (Status-Line). Она состоит из версии протокола (HTTP-Version), числового кода состояния (Status-Code) и поясняющей фразы (Reason-Phrase),

разделенных символами SP(пробел). CR (возврат каретки) и LF(перевод строки) не допустимы в

Status-Line, за исключением конечной последовательности CRLF.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF.

	Status-Code	**Reason-Phrase**
1	**100**	Продолжать, Continue
2	**101**	Переключение протоколов, ; Switching Protocols
3	**200**	OK
4	**201**	Создан, Created
5	**202**	Принято, Accepted
6	**203**	Не авторская информация,; Non-Authoritative Information
7	**204**	Нет содержимого, No Content
8	**205**	Сбросить содержимое, Reset; Content
9	**206**	Частичное содержимое, Partial; Content
10	**300**	Множественный выбор, Multiple Choices
11	**301**	Постоянно перенесен, Moved Permanently
12	**302**	Временно перемещен, Moved Temporarily
13	**303**	Смотреть другой,See Other
14	**304**	Не модифицирован, Not Modified
15	**305**	Используйте прокси-сервер, Use Proxy
16	**400**	Испорченный Запрос, Bad Request
17	**401**	Несанкционированно, Unauthorized
18	**402**	Требуется оплата, Payment Required
19	**403**	Запрещено, Forbidden
20	**404**	Не найден, Not Found
21	**405**	Метод не дозволен, Method Not Allowed
22	**406**	Не приемлем, Not Acceptable
23	**407**	Требуется установление подлинности через прокси-сервер, Proxy Authentication Required

24	**408**	Истекло время ожидания запроса, Request Timeout
25	**409**	Конфликт, Conflict
26	**410**	Удален, Gone
27	**411**	Требуется длина, Length Required
28	**412**	Предусловие неверно,
29	**413**	Объект запроса слишком большой, Request Entity Too Large
30	**414**	URI запроса слишком длинный, Request-URI Too Long
31	**415**	Неподдерживаемый медиа тип, Unsupported Media Type
32	**500**	Внутренняя ошибка сервера, Internal Server Error
33	**501**	Не реализовано, Not Implemented
34	**502**	Ошибка шлюза, Bad Gateway
35	**503**	Сервис недоступен, ServiceUnavailable
36	**504**	Истекло время ожидания от шлюза, Gateway Timeout
37	**505**	Не поддерживаемая версия HTTP, HTTP Version Not Supported

Клиент или сервер МОГУТ передать объект (сообщение). Объект состоит из полей заголовка объекта (entity-header) и тела объекта (entity-body), хотя некоторые ответы могут включать только заголовки объекта (entity-headers). Объект может посылаться и клиентом, и сервером. Тело объекта (если оно присутствует) посылается с HTTP запросом или ответом и имеет формат и кодирование, определяемое полями заголовка объекта (entity-header fields). Тело объекта (entity-body) представлено в сообщении только тогда, когда присутствует тело сообщения (message-body). Тело объекта (entity-body) получается из тела сообщения (message-body) декодированием, указанным в поле Transfer-Encoding. Тип данных этого тела определяется полями заголовка Content-Type и Content-Encoding. Они определяют двухуровневую упорядоченную модель кодирования: entity-body:= Content-Encoding(Content-Type(data)).Тип содержимого (Content-Type) определяет медиа тип основных данных (текст, изображение, другое). Кодирование содержимого (Content-Encoding) может использоваться для указания любого дополнительного кодирования содержимого, примененного к данным (обычно с целью сжатия данных). Кодирование содержимого (Content-Encoding) является свойством запрошенного ресурса. По умолчанию никакого кодирования не задано. В любое HTTP/1.1 сообщение, содержащее тело объекта (entity-body). В том случае, когда медиа тип не представлен полем Content-Type, получатель МОЖЕТ попытаться предположить медиа тип, проверяя содержимое и/или расширение (расширения) в имени URL, используемого для идентификации ресурса. Если медиа тип остался

нераспознан, получателю СЛЕДУЕТ обрабатывать его как тип "application/octet-stream".

Пример HTTP-запроса:

```
GET /default.aspx HTTP/1.1
```

Пример HTTP-ответа:

```
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
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 страница в HTML)

Более подробно об HTTP-протоколе см.
http://ru.wikipedia.org/wiki/HTTP
http://tools.ietf.org/html/rfc1945
http://tools.ietf.org/html/rfc2616

Классы для работы с HTTP

Платформа .NET сильно облегчила взаимодействие с HTTP-сервером, предоставив несколько высокоуровневых классов пространства имен System.Net для поддержки протокола, которые дают возможность управлять заголовками и соединениями, выполнять предварительную аутентификацию (определение личности клиента), шифрование, поддержку работы с прокси-сервером, конвейерную обработку:

	Класс	Пространство имен	Базовый класс	Описание
1	**HttpWebRequest**	System.Net	WebRequest	HTTP-запрос
2	**HttpWebResponse**	System.Net	WebResponse	HTTP-ответ
3	**WebClient**	System.Net	Component	Простые методы получения и отправки данных для URI
4	**ServicePoint**	System.Net	Object	Обработка соединения с URI

5	**ServicePointManager**	System.Net	Object	Управляет объектами ServicePoint
6	**Uri**	System	MarshalByRef Object	Легкое управление URI
7	**UriBuilder**	System	Object	Создание и модификация объектов URI

Отправка запросов с использованием класса HttpWebRequest

HttpWebRequest предоставляет возможность отправить запрос . Создаем и отсылаем запрос:

```
HttpWebRequest reqw=(HttpWebRequest)HttpWebRequest.Create(«http://itstep.org»);
reqw.GetResponse();
```

Получение ответов с использованием класса HttpWebResponse

Однако отсылка запроса без получения результата – достаточно бессмысленное занятие. Улучшим программу - выведем на экран содержимое ресурса(HTML-странички)

```
HttpWebRequest reqw=(HttpWebRequest)HttpWebRequest.Create(«http://itstep.org»);
HttpWebResponse resp= reqw.GetResponse();// создаем объект отклика
StreamReader sr=new StreamReader(resp.GetResponseStream(),Encoding.Default);//создаем поток для чтения отклика
Console.WriteLine(sr.ReadToEnd()); // вывести на экран все, что читается
        sr.Close();
```

Установка заголовков запроса

Заголовки запроса нам позволят более тонко настроить взаимодействие с HTTP-сервером, например, установить язык по умолчанию

```
HttpWebRequest reqw=(HttpWebRequest)HttpWebRequest.Create(«http://itstep.org»);
reqw.Headers.Add(“Accept-Language: ru-ru”);//установка русского языка по умолчанию
```

Кроме этого для установки значений некоторых заголовков можно воспользоваться свойствами класса:

	Свойство	Тип данных	HTTP-заголовок

1	**Accept**	String	Accept
2	**Connection**	string	Connection
3	**ContentLength**	Long	Content-Length
4	**Expect**	String	Expect
5	**IfModifiedSince**	DateTime	If-Modified-Since
6	**Referer**	String	Referer
7	**TransferEncoding**	String	Transfer-Encoding
8	**ContentType**	String	Content-Type
9	**UserAgent**	String	User-Agent

Установить эти заголовки через reqw.Headers.Add **нельзя**.

Чтение заголовков ответа.

Чтение заголовков доступно из коллекции Headers или из других свойств класса HttpWebResponse , указанных в таблице:

	Свойство	Тип данных	HTTP-Заголовок
1	ContentEncoding	String	ContentEncoding
2	ContentLength	Long	ContentLength
3	ContentType	String	ContentType
4	LastModified	DateTime	LastModified
5	Server	String	Server

Чтение из коллекции :

```
    HttpWebResponse resp= reqw.GetResponse();
  foreach( string header in resp.Headers)
    Console.WriteLine("{0}:{1}",header,resp.Headers[header]);
```

Использование класса WebClient

Класс WebClient упрощает обмен данными с сервером. Для получения данных используется метод DownloadData()

```
    // создание объекта web-клиент
    WebClient client= new WebClient();
    // получение содержимого странички
    byte[] urlData = clientDownLoadData("http://www.yandex.ru");
```

```
// преобразование полученного содержимого в строку для отображения в консоли
string page = Encoding.ASCII.GetString(urlData);
Console.WriteLine(page);
```

Для получения файла с сервера используется метод DownloadFile()

```
// создание объекта web-клиент
WebClient client= new WebClient();
string fileCopy = “c:\\ttt.gif”, urlString=”http://www.yandex.ru/images/point.gif”;
// закачка web-ресурса  в файл с именем fileCopy
client.DownloadFile(urlString,fileCopy);
```

Для считывания данных частями используется метод OpenRead(), получающий поток, доступный для чтения:

```
// создание объекта web-клиент
WebClient client= new WebClient();
string fileCopy = “c:\\ttt.gif”, urlString=”http://www.yandex.ru/images/point.gif”;
// связываем URL с потоком чтения
Stream copy=client.OpenRead(urlString);
// создаем поток для хранения закачанных данных
FileInfo fi=new FileInfo(fileCopy);
StreamWriter sw=fi.CreateText();
// закачка web-ресурса  в файл с именем fileCopy
sw.WriteLine(copy.ReadToEnd());
sw.Close();
copy.Close();
```

Для передачи данных серверу используется метод OpenWrite(), который возвращает поток, доступный для записи (используется метод HTTP-передачи POST):

```
// создание объекта web-клиент
WebClient client= new WebClient();
string TextToUpload = “User=Vasia&passwd=okna”,
urlString=”http://www.yandex.ru/page22.aspx”;
// Преобразуем текст в массив байтов
byte[] uploadData=Encoding.ASCII.GetBytes(TextToUpload);
// связываем URL с потоком записи
Stream upload=client.OpenWrite(urlString,”POST”);
// загружаем данные на сервер
upload.Write(uploadData,0,uploadData.Length);
upload.Close();
```

Для передачи данных серверу другими HTTP-методами используется метод UploadData()

```
// создание объекта web-клиент
```

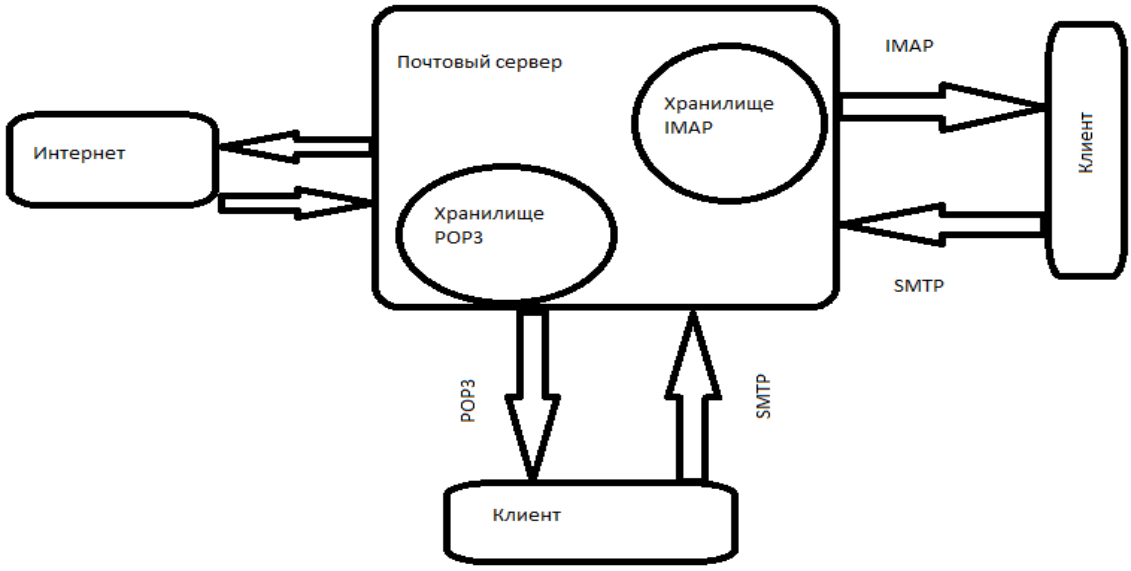
```
        WebClient client= new WebClient();
        client.Credentials = System.Net.CredentialCashe.DefaultCredentials;
          // добавляем HTTP-заголовок
         client.Headers.Add(“Content-Type”, “application/x-www-form-urlencoded”);
       string TextToUpload = “User=Vasia&passwd=okna”,
urlString=”http://www.yandex.ru/page22.aspx”;
       // Преобразуем текст в массив байтов
       byte[] uploadData=Encoding.ASCII.GetBytes(TextToUpload);
       // копируем данные методом GET
       byte[] respText=client.UploadData(urlString,”GET”,uploadData);
       // загружаем данные на сервер
       upload.Write(uploadData,0,uploadData.Length);
       upload.Close();
```

Работа с электронной почтой

Общий обзор почтовых протоколов SMTP, POP3, IMAP

Отправка и получение электронной почты в настоящее время осуществляется при помощи почтовых протоколов, которые используют TCP/IP в качестве транспортного протокола. Для отправки электронной почты используют почтовый протокол SMTP, для получения – POP3. Существующий протокол IMAP справляется с обеими задачами, но чаще его используют вместо POP3 .

Схема отправки и получения почты



Протокол SMTP

Протокол SMTP (Simple Mail Transport Protocol) определяет взаимодействие между серверами, транспортирующими электронную почту и между клиентом и сервером(RFC 2821). Взаимодействие производится путем передачи ограниченного списка команд, за которыми следуют данные.

SMTP-сервером называется программа, ожидающая запроса от клиента на передачу почты или от другого SMTP-сервера на получение. SMTP-клиент – это программа, обращающаяся к серверу для передачи ему почтового сообщения. **SMTP-сервер** хранит базу зарегистрированных пользователей и их паролей. Таким образом, невозможна передача почтовых сообщений от незарегистрированных пользователей.

Сводка некоторых команд **SMTP-сервера**:

	Команда (и ее 4-буквенный вариант)	Описание
	HELLO (HELO)	Идентификация SMTP-клиента
	MAIL (MAIL)	Инициирует почтовую транзакцию (доставка почты) в почтовые ящики
	RECEPIENT (RCPT)	Идентификация получателя. Можно несколько команд за сеанс
	DATA (DATA)	Начало почтовых данных
	SEND (SEND)	Инициирует почтовую транзакцию на терминалы (устаревшее)
	SEND или MAIL (SOML)	Если получатель активен – отправка почты на терминал, иначе – в почтовый ящик
	SEND и MAIL (SAML)	Получает почту на терминал и в почтовый ящик
	RESET (RSET)	Прерывает текущую почтовую транзакцию
	WERIFY (WRFY)	Требует от приемника подтвердить, что ее аргумент является действительным именем пользователя.
	EXPAND (EXPN)	Команда SMTP-приемнику подтвердить, действительно ли аргумент является адресом почтовой рассылки и если да, вернуть адрес получателя

		сообщения
	HELP (HELP)	Команда SMTP-приемнику вернуть сообщение-справку о его командах
	NOOP (NOOP)	Требует от получателя не предпринимать никаких действий, а только выдать ответ OK. Используется главным образом для тестирования
	QUIT (QUIT)	Требует выдать ответ OK и закрыть текущее соединение
	TURN (TURN)	Команда SMTP-приемнику либо сказать OK и поменяться ролями, то есть стать STMP- передатчиком, либо послать сообщение-отказ и остаться в роли SMTP-приемника

Классы .Net для работы SMTP

Пространство System.Net.Mail

Пространство имен System.Net.Mail содержит 3 класса и 3 перечисления:

	Класс	**Описание**
1	**MailMessage**	Сообщение электронной почты
2	**SmtpClient**	Реализует отправку MailMessage через SMTP
3	**MailAttacment**	Вложения в почтовое сообщение
	Перечисление	**Описание**
1	**MailEncoding**	Тип кодирования Base46 или UUEncode
2	**MailFormat**	Формат сообщения Text или HTML
3	**MailPriopity**	High, Medium, Low – приоритет сообщения

Класс MailMessage

Основной класс для формирования сообщения. Некоторые члены класса:

	Свойство	Описание
1	**Attachments**	Коллекция вложений
2	**Bcc**	Список для рассылки копий в виде строки почтовых адресов, разделенных _;_ (Blind Carbon Copy)
3	**Body**	Тело сообщения
4	**BodyFormat**	Формат электронной почты

		MailFormat.Text или MailFormat.Html
5	**Cc**	Список для рассылки копий в виде строки почтовых адресов, разделенных _;_ (Carbon Copy)
6	**From**	Почтовый адрес отправителя
7	**Subject**	Тема сообщения
8	**To**	Почтовый адрес получателя
9	**Priority**	Приоритет сообщения(перечисление MailPriority)
10	**UrlContentBase**	HTTP-заголовок Content-Base. База для всех относительных Url
11	**UrlContentLocation**	HTTP-заголовок Content-Location
12	**Headers**	Список нестандартных заголовков, передаваемых с сообщением

Создание почтового сообщения :

```
MailMessage post=new MailMessage();
post.From = «vasily@pupkin.com»;
post.To= «Lusi@pupkin.com»;
post.Subject= «Test message»;
post l.BodyFormat  = MailFormat.Text;
post.Body = «post message»;
```

Такое почтовое сообщение, в принципе, может быть отправлено

Класс Attachment

Этот класс предназначен для создания объекта вложения, который впоследствии присоединяется к **MailMessage.Attachments**.

Класс имеет 2 важных с точки зрения отправки почты свойства: Attachment.Filename – имя присоединяемого файла, **Attachment.Encoding** – тип кодировки вложения(**MailEncoding.Base64** и **MailEncoding.UUEncode**). Пример создания почтового сообщения с вложением:

```
MailMessage post=new MailMessage();
post.From = «vasily@pupkin.com»;
post.To= «Lusi@pupkin.com»;
post.Subject= «Spring Calndar»;
post l.BodyFormat  = MailFormat.Text;
post.Body = «1 April»;
MailAttachment at=new MailAttachment();
at.Filename= «C:\MyHohma.Jpg»;
post.Attachments.Add(at);
```

Класс SmtpClient

Предназначен для отправки почтового сообщения SMTP-серверу.
Основные свойства и методы:

	Метод	**Назначение**
	Send	Отправка почты
	SendAsync	Неблокирующая отправка почты
	SendAsyncCancel	Завершение операции неблокирующей отправки почты
	Свойство	**Описание**
	EnableSsl	Использовать шифрование
	Host	Строка- адрес сервера
	Port	Целое число – номер порта
	TimeOut	Время ожидания завершения команды Send
	Событие	**Описание**
	SendCompleted	Завершение асинхронной операции отправки почты

Пример отправки почты

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net.Mail;


namespace Mailer
{
   class Program
   {
      string prompt(string query)
      {
         Console.WriteLine(query);
         return Console.ReadLine();
      }
      static void Main(string[] args)
      {
         Program app = new Program();
         // заполняем поля почтового сообщения
         app.Dialog();
         //  пытаемся отправить сообщение
         app.SendMail();
      }
      string to;
      string from;
      string subject;
```

```
        string body;
        string server;

        void Dialog()
        {
            to = prompt("Введите адрес получателя:");
            from = prompt("Введите адрес отправителя:");
            subject = prompt("Введите тему");
            body = prompt("Введите текст сообщения:");
            server = prompt("Введите адрес сервера:");
        }

        public void SendMail()
        {
            MailMessage message = new MailMessage(from, to, subject, body);
            SmtpClient client = new SmtpClient(server);
            Console.WriteLine("Сосчитайте до 100");
            client.Timeout = 10000;// устанавливаем TimeOut 10000 milliseconds

            client.UseDefaultCredentials = true;
            try
            {
                client.Send(message);
                Console.WriteLine("Сообщение отправлено");
            }
            catch (SmtpException se)
            {
                Console.WriteLine("Сообщение не отправлено по причине "+se.Message);
            }
        }

    }
}
```

Использование FTP

Общий обзор

FTP означает интерфейс пользователя, реализующий ARPANET стандартный протокол передачи файлов. Эта программа позволяет пользователю передавать файлы между двумя компьютерами, связанными между собой локальной (LAN) или глобальной (WAN) сетью. При этом компьютерные платформы могут быть различных типов. В этом и заключается главная особенность FTP в сети.

Если ваша система имеет FTP и подсоединена к Internet, то вы получите доступ к огромному числу архивов, хранящихся на других системах.

Терминология FTP

При работе с FTP используется модель клиент-сервер. Клиент передает запросы и получает ответы по двум портам TCP- соединения(20 – порт передачи данных и 21- порт передачи команд). Некоторые FTP-команды:

dir [удаленная_директория] [локальный_файл]

ls [удаленная_директория] [локальный_файл]

Выводит список файлов в директории либо на стандартный вывод, либо, если указано имя локального файла, в этот файл.

get [удаленный_файл] [локальный_файл]

Вызывает передачу копии удаленного файла на ваш компьютер. В случае если имя локального файла не было задано, то оно совпадает с именем удаленного файла.

mget [удаленные_файлы]

Для получения нескольких файлов

hash

Служит переключателем для индикации каждого полученного блока данных в 1024 байта, повышает наглядность процедуры.

cd [удаленная_директория]

Сменить директорию. Существуют также 'cdup' или 'cd' для возврата на один или выше

lcd

Меняет рабочую директорию на локальной машине (без аргумента - переход в домашнюю директорию пользователя)

bin (или binary)

Переключает в режим передачи двоичных файлов

ascii

Переключает в режим передачи текстовых файлов (обычно по умолчанию).

prompt

Переключает интерактивную подсказку. Часто при использовании команды 'mget' желательно предварительно набрать 'prompt', чтобы не давать многократные подтверждения.

pwd

Выводит имя удаленной рабочей директории.

mkdir [имя_директории]

Создает директорию на удаленной машине

open хост [порт]

Устанавливает соединение с заданным FTP сервером

put [локальный_файл] [удаленный_файл]

Пересылает файл на удаленную систему. Если имя удаленного файла не указано, то оно совпадает с именем на локальной системе.

quit

Синоним для 'bye'

recv [удаленный_файл] [локальный_файл]

Синоним для команды 'get'

reget [удаленный_файл] [локальный_файл]

"Дополучение" удаленного файла в том случае, когда часть его уже есть на локальной машине. Команда особенно полезна для получения больших файлов при возможных резервах соединения.

delete [удаленный_файл]

Стирает удаленный файл

close

Обрывает FTP сеанс с удаленным сервером и возвращает к командному интерпретатору

bye

Оканчивает работу с FTP сервером и приводит к выходу и из интерпретатора.

Пример типичной FTP сессии

```
220 FTP server ready.
USER ftp //Анонимус
230 Login successful.
PASV
227 Entering Passive Mode (192,168,254,253,233,92)//Клиент должен открыть соединение на переданный  IP
LIST
150 Here comes the directory listing. //Сервер передает список файлов в директории
226 Directory send OK.
CWD incoming
250 Directory successfully changed.
PASV
227 Entering Passive Mode (192,168,254,253,207,56)
STOR gyuyfotry.avi
150 Ok to send data. //Клиент передает содержимое файла
226 File receive OK.
QUIT
221 Goodbye.
```

Классы для работы с FTP

System.Object
  System.MarshalByRefObject
    System.Net.WebRequest
      System.Net.FtpWebRequest

FtpWebRequest – класс для связи с FTP-сервером. ри помощи методов этого класса инициируется передача FTP-команд на сервер.

Основные члены класса:

	Метод	Описание
	WebRequest.Create	Статический метод для создания объекта

		FtpWebRequest
	GetResponce	Получает ответ FTP-сервера в виде FtpWebResponce
	GetRequestStream	Получает поток для выгрузки данных на сервер(upload)
	BeginGetResponce	Начало асинхронной передачи запроса серверу и получения ответа
	EndGetResponce	Завершение асинхронной операции получения ответа сервера
	BeginGetRequestStream	Асинхронное открытие потока для выгрузки на сервер
	EndGetRequestStream	Завершение асинхронной операции получения потока для выгрузки на сервер
	Свойство	**Описание**
	Method	Возвращает или задает FTP-команду
	TimeOut	Интервал ожидания синхронной операции в миллисекундах
	UsePassive	Пассивный или активный режим передачи
	UseBinary	Тип данных при передаче
	RenameTo	Имя переименовываемого файла
	ReadWriteTimeOut	Время ожидания операции чтения или записи
	Proxy	Определение прокси-сервера
	EnableSsl	Использование шифрования при передаче
	ContentType	Тип содержимого
	Credentials	Данные для связи с сервером
	ContentOffset	Смещение загружаемого файла

System.Object
  System.MarshalByRefObject
    System.NetWebResponse
      System.Net.FtpWebResponse

Класс **FtpWebResponse** получает ответ сервера и используется для обработки результатов этого ответа
Основные члены класса:

	Методы	**Назначение**
	GetResponseStream	Получает поток для чтения файла или выгрузки на сервер
	Close	Освобождает ресурсы
	Свойства	Описание
	ContentType	Тип содержимого
	ContentLength	Длина файла
	BannerMessage	Предварительное сообщение сервера

	ExitMessage	Сообщение по завершении FTP-сеанса
	WelcomeMessage	Сообщение сервера после подключения
	LastModified	Дата последнего изменения файла
	StatusCode	Текущее состояние FTP-сессии
	StatusDescription	Описание текущего состояния FTP-сессии
	IsFromCache	был ли этот ответ получен из кэша

Пример использования FTP

Удаление файла на сервере

```
string FileName=”ftp://glamurconcurs.com/lucie.jpg”;
// создаем объект FTP- запроса
FtpWebRequest request = (FtpWebRequest)WebRequest.Create(FileName);
// задаем FTP-команду
request.Method = WebRequestMethods.Ftp.DeleteFile;
// выполняем запрос и получаем результат
FtpWebResponse response = (FtpWebResponse) request.GetResponse();
// обрабатываем результат
Console.WriteLine("Delete status: {0}",response.StatusDescription);
// закрываем сессию
response.Close();
```

Варианты экзаменационных заданий (выберите одно):

1. Построить дерево сайта. Ссылки на другие сайты ограничить 1 уровнем. Учесть возможность рекурсивных ссылок.

2. Закачать сайт. Закачки по ссылкам на другой сайт не производить. Учесть возможность рекурсивных ссылок.

3. Написать чат (программу обмена сообщений между несколькими пользователями) на базе сокетов с выделенным сервером

4. Написать чат (программу обмена сообщений между несколькими пользователями) на базе дейтаграмм

5. Написать программу сетевой игры в шашки с возможностью подключения болельщиков.

6. Программа закачки с FTP-сервера с возможностью продолжения прерванной закачки и организовать проверку через определенный интервал времени изменений закачанных файлов на сервере (обновление)

7. Написать сервер удаленного доступа нескольких клиентов для базы MS ACCESS и простейший клиент с редактором SQL-запросов.