top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ В NET FRAMEWORK

TCP И UDP СОКЕТЫ, UNICAST, BROADCAST, MULTICAST

ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ПРОТОКОЛОВ HTTP, SMTP, FTP

Урок №6

Использование сетевых протоколов HTTP, SMTP, FTP

Содержание

1. Обзор HTTP протокола

HTTP — Протокол прикладного уровня поверх TCP/IP, используемый для передачи гипертекста в WWW и локальных сетях. В основном используется для передачи содержимого веб-страниц. 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 сообщени.

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.

Таблица 1. Названия заголовков HTTP-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	Атрибуты авторизации пользователя
6	From	Адрес клиента
7	Host	Имя хоста, с которого запрашивается ресурс
8	If-Modified-Since	Действие в случае измененного содержимого
9	If-Match	Действие в случае совпадения
10	If-None-Match	Действие в случае несовпадения
11	If-Range	Действие в случае превышения размеров
12	If-Unmodified-Since	Действие в случае неизменяемого содержимого
13	Max-Forwards	Максимальное количества ссылок
14	Proxy-Authorization	Атрибуты авторизации пользователя на прокси
15	Range	Размеры
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.
```

Таблица 2. Коды ответов HTTP-сервера

№	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

№	Status-Code	Reason-Phrase
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 кодирова-

ние, 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».

1.1. Пример HTTP-запроса

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

1.2. 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.

2. 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:

Таблица 3. Классы .Net для работы с HTTP

№	Класс	Пространство имен	Базовый класс	Описание
1	HttpWebRequest	System.Net	WebRequest	HTTP-запрос
2	HttpWebResponce	System.Net	WebResponce	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	ServicePoint Manager	System.Net	Object	Управляет объектами ServicePoint
6	Uri	System	MarshalByRef Object	Легкое управление URI
7	UriBuilder	System	Object	Создание и модификация объектов URI

2.1. 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();
```

2.2. 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= (HttpWebResponse) 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();
```

2.3. 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:

Таблица 4. Свойства класса WebRequest

№	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 заголовки через reqw.Headers.Add *нельзя*.

2.4. Чтение заголовков ответа

Чтение заголовков доступно из коллекции Headers или из других свойств класса HttpWebResponse, указанных в таблице 5.

Чтение из заголовков из коллекции:

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

Таблица 5. Свойства класса WebResponse

№	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

2.5. Использование класса WebClient

Класс WebClient упрощает обмен данными с сервером. Для получения данных используется метод Download-Data()

```
//создание объекта web-клиент
WebClient client= new WebClient();
//получение содержимого странички
byte[] urlData =
       client.DownLoadData("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();
```

Пример работы классов HttpRequest и HttpResponce:

```
namespace HTTP_Client
{
    public partial class Form1 : Form
    {
```

```
        public Form1()
        {
            InitializeComponent();
        }

        private void sendButton_Click(object sender,
                                      EventArgs e)
        {
            HttpWebRequest req = (HttpWebRequest)
                HttpWebRequest.Create(URL.Text);
            req.Method = "GET";
            if (IfProxy.Checked)
            {
                WebProxy proxy =
                          new WebProxy(proxyAddr.Text);

                proxy.Credentials =
                    new NetworkCredential(proxyUser.
                                Text,
                                proxyPassword.Text);
                req.Proxy = proxy;
            }
            HttpWebResponse rez = (HttpWebResponse)
                    req.GetResponse();
            StreamReader sr = new StreamReader(
                    rez.GetResponseStream(),
                    Encoding.Default);
            response.Text = sr.ReadToEnd();
        }
    }
}
```

В результате получаем приложение, читающее html-страницу и отображающую ее в текстовом редакторе в виде текста:

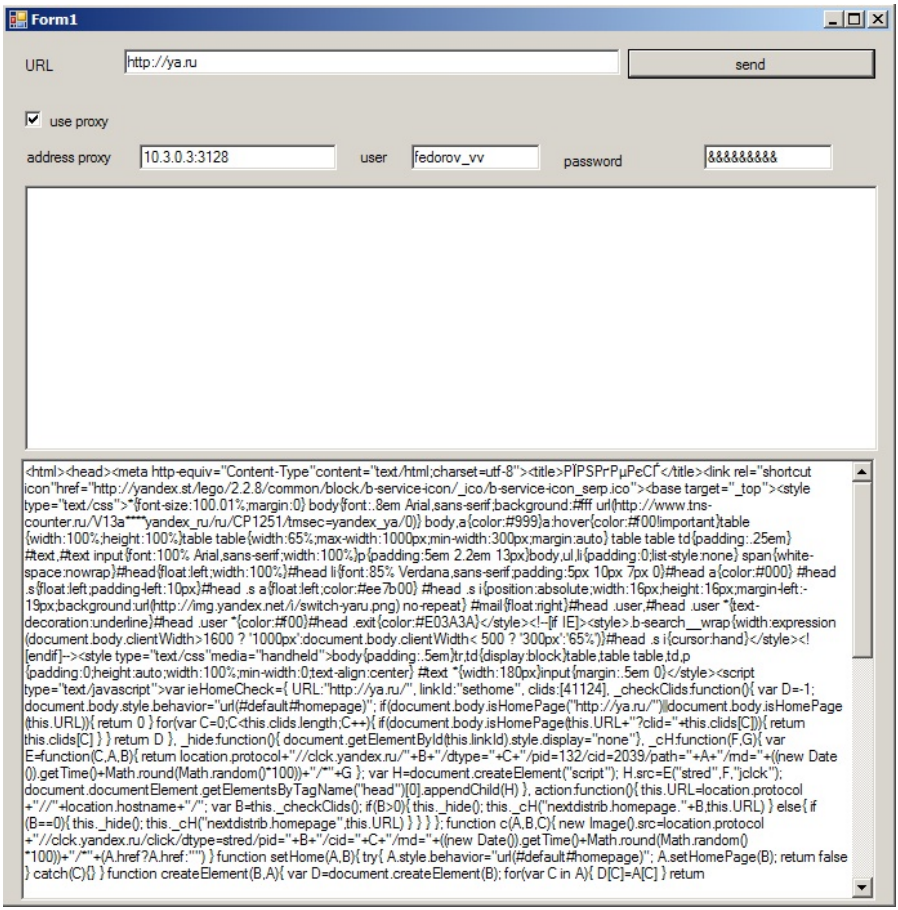
Form1
URL
http://ya.ru
send
use proxy
address proxy
10.3.0.3:3128
user
fedorov_vv
password
&&&&&&&&

Рисунок 1

3. Работа с электронной почтой

3.1. 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 потокола. 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

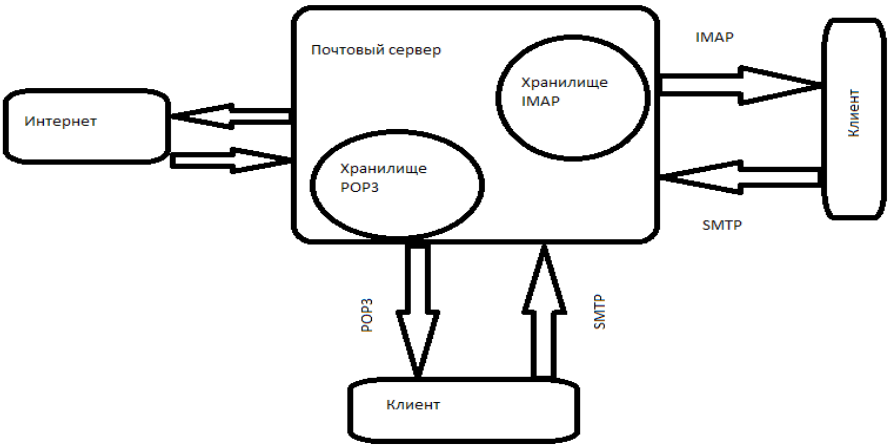

Рисунок 2

3.2. Протокол SMTP

Общий обзор

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.

Терминология SMTP

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. **Почтовый ящик** (*mailbox*) — это область памяти (чаще всего — каталог) на сервере, где хранятся принятые письма, предназначенные некоторому клиенту. **Почтовое сообщение** — это текстовый блок, в котором содержатся заголовки, тело сообщения и вложение (одно или несколько). **Заголовки** сообщения определяют получателя, отправителя, почтовый адрес, тип содержимого, наличие вложений, тип кодирования, кодировку текста, вид сообщения (текст-html) и др. **Вложение** — это дополнительный файл, который может иметь содержимое любого типа(изображение, звуковой файл, видео, программу…).

Команды SMTP

Команды отсылаются SMTP-серверу в текстовом виде. Сначала идет текстовая команда, затем ее параметры. Ответ сервера сопровождается кодом ошибки, за которым следует разъяснение. Сеанс обмена может выглядеть так:

```
open mail.olgs.gov 25
Trying...Connect to mail.olgs.gov
220 mail.olgs.gov - Server ESMTP (PMDF V4.3-10 #2381)

hello mysite.com
250 mail.olgs.gov OK, mysite.com

mail from:<habitant@mysite.com>
250 Address Ok.

rcpt to:<kravchuk@ mail.olgs.gov>

250 kravchuk@ mail.olgs.gov OK

data
354 Enter mail, end with a single ....
SUBJECT:E-mail chapter
Ljonja, thanks for the live
.
250 OK

quit
221 Bye received. Goobye
```

Таблица 5. Сводка некоторых команд SMTP-сервера

№	Команда (и ее 4-буквенный вариант)	Описание
1	HELLO (HELO)	Идентификация SMTP-клиента
2	MAIL (MAIL)	Инициирует почтовую транзакцию (доставка почты) в почтовые ящики

№	Команда (и ее 4-буквенный вариант)	Описание
3	RECEPIENT (RCPT)	Идентификация получателя. Можно несколько команд за сеанс
4	DATA (DATA)	Начало почтовых данных
5	SEND (SEND)	Инициирует почтовую транзакцию на терминалы (устаревшее)
6	SEND или MAIL (SOML)	Если получатель активен — отправка почты на терминал, иначе — в почтовый ящик
7	SEND и MAIL (SAML)	Получает почту на терминал и в почтовый ящик
8	RESET (RSET)	Прерывает текущую почтовую транзакцию
9	WERIFY (WRFY)	Требует от приемника подтвердить, что ее аргумент является действительным именем пользователя.
10	EXPAND (EXPN)	Команда SMTP-приемнику подтвердить, действительно ли аргумент является адресом почтовой рассылки и если да, вернуть адрес получателя сообщения
11	HELP (HELP)	Команда SMTP-приемнику вернуть сообщение-справку о его командах
12	NOOP (NOOP)	Требует от получателя не предпринимать никаких действий, а только выдать ответ OK. Используется главным образом для тестирования
13	QUIT (QUIT)	Требует выдать ответ OK и закрыть текущее соединение
14	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

Используя команды обмена можно написать smtp-клиент самостоятельно.

MS.NET Framework предоставляет удобный набор классов, чтобы можно было легко встраивать обмен почтой в свои приложения.

3.3. Классы .Net для работы SMTP

.Net содержит набор классов для создания SMTP-клиента и не содержит классов для создания SMTP-сервера.

Сначала опишем порядок отправки сообщения с использованием классов .Net.

Первым делом создается сообщение, затем создается класс клиента, сообщение присоединяется к классу клиента и клиент отправляет почтовое сообщение. Если необходимо добавить вложения, то создаются экземпляры вложений, которые добавляются в экземпляр сообщения. Потом полученный контейнер уже добавляется к клиенту. Если имеются вложения, информация об этом будет содержаться в заголовке. Для выполнения всех действий с почтой требуются пространства System.Net и System.Net.Mail.

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

Таблица 6. Пространство System.Net.Mail

№	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

Рассмотрим основные классы из пространства имен System.Net.Mail и создадим почтовое сообщение без вложения и с вложением.

3.4. Класс MailMessage

Основной класс для формирования сообщения. Он содержит заголовки почтового сообщения, тело сообщения и коллекцию файлов вложений. К заголовкам относятся адрес получателя, адрес отправителя, дата сообщения, адрес копии, адрес перенаправления, адрес скрытой ссылки и др.

Таблица 7. Некоторые члены класса MailMessage

№	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	UrlContent Location	HTTP-заголовок Content-Location
12	Headers	Список нестандартных заголовков, передаваемых с сообщением

Создание почтового сообщения. Такое почтовое сообщение, в принципе, может быть отправлено.

```
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";
```

3.5. Класс Attachment

Этот класс предназначен для создания объекта вложения, который впоследствии присоединяется к коллекции 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);
```

3.6. Класс SmtpClient

Предназначен для отправки почтового сообщения SMTP-серверу. Содержит средства диагностики успешной

отправки и асинхронной отправки писем. Не позволяет узнать о дальнейшей судьбе письма (дошло ли письмо до получателя). Имеется возможность шифрования сообщения, используя протокол SSL (*Secure Socket Level*).

Таблица 8. Основные свойства и методы класса SmtpClient

№	Метод	Назначение
1	Send	Отправка почты
2	SendAsync	Неблокирующая отправка почты
3	SendAsyncCancel	Завершение операции неблокирующей отправки почты
4	Свойство	Описание
5	EnableSsl	Использовать шифрование
6	Host	Строка — адрес сервера
7	Port	Целое число — номер порта
8	TimeOut	Время ожидания завершения команды Send
9	Событие	Описание
10	SendCompleted	Завершение асинхронной операции отправки почты

3.7. 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);
            }
        }
    }
}
```

В случае использования многофункционального оконного приложения нежелательно заставлять пользователя ожидать отправки, лучше предложить ему

Вариант асинхронной отправки почты (обязательно поменяйте адрес сервера, логин и пароль в NetworkCredential):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
using System.Net.Mail;
using System.Net;

namespace MailSender
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        string server="100.30.0.30";//здесь задается
                                    //адрес сервера
        private void button2_Click(object sender,
                                   EventArgs e)
        {
            //создаем объект сообщения
            MailMessage message = new MailMessage(
                        fromBox.Text, toBox.Text,
                        themeBox.Text, bodyBox.Text);
            //создаем объект отправки
            SmtpClient client = new SmtpClient(server);
            client.Port = 25; //здесь устанавливается
                              //порт сервера
            //настройки для отправки почты(логин и пароль
            client.Credentials =
                 new NetworkCredential("test","test");
            //вызываем асинхронную отправку сообщения
            client.SendAsync(message,"That's all");
        }
    }
}
```

Внешний вид приложения (само приложение находится в папке *source*).

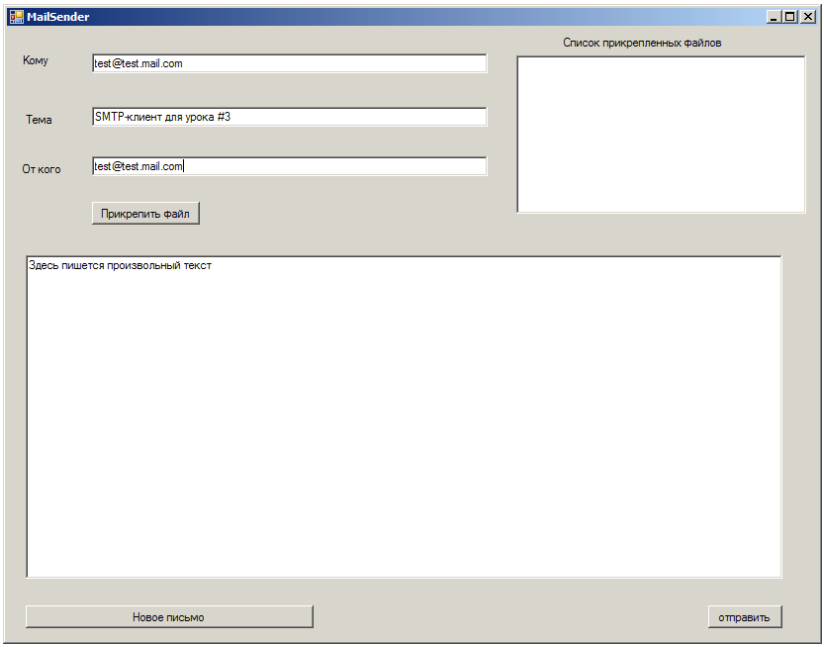

Рисунок 3

3.8. Протокол POP3

POP3 (*Post Office Protocol*) — определяет взаимодействие между сервером и клиентом для получения почты клиентом. Сервер ожидает соединения с клиентом, производит авторизацию, выполняет команды клиента по работе с его почтовым ящиком и завершающие операции(сессия): «POP3 сессия состоит из нескольких режимов. Как только соединение с сервером было установлено и сервер отправил приглашение, то сессия переходит в режим AUTHORIZATION (*Авторизация*). В этом режиме клиент должен идентифицировать себя на сервере. После успешной идентификации сессия переходит

в режим TRANSACTION (*Передача*). В этом режиме клиент запрашивает сервер выполнить определённые команды. Когда клиент отправляет команду QUIT, сессия переходит в режим UPDATE. В этом режиме POP3 сервер освобождает все занятые ресурсы и завершает работу». (*По материалам* www.codenet.ru).

Пример типичной сессии POP3

```
S: +OK POP3 server ready
C: USER MonstrVB
S: +OK MonstrVB is a real hoopy frood
C: PASS mymail
S: +OK MonstrVB's maildrop has 2 messages (320 octets)
C: QUIT
S: +OK dewey POP3 server signing off
```

Таблица 8. Команды протокола

Формат команды	Описание
APOP [имя] [digest]	Команда служит для передачи серверу имени пользователя и зашифрованного пароля (digest).
USER [имя]	Передаёт серверу имя пользователя.
PASS [пароль]	Передаёт серверу пароль почтового ящика
DELE [N сообщения]	Сервер помечает указанное сообщение для удаления. Сообщения, помеченные на удаление, реально удаляются только после закрытия транзакции (закрытие транзакций происходит обычно после посыла команды QUIT, кроме этого, например, на серверах закрытие транзакций может происходить по истечении определённого времени, установленного сервером).

Формат команды	Описание
LIST [N сообщения]	Если был передан аргумент, то сервер выдаёт информацию об указанном сообщении. Если аргумент не был передан, то сервер выдаёт информацию обо всех сообщениях, находящихся в почтовом ящике. Сообщения, помеченные для удаления, не перечисляются.
NOOP	Сервер ничего не делает, всегда отвечает положительно.
RETR [N сообщения]	Сервер передаёт сообщение с указанным номером.
RSET	Этой командой производится откат транзакций внутри сессии. Например, если пользователь случайно пометил на удаление какие-либо сообщения, он может убрать эти пометки, отправив эту команду.
STAT	Сервер возвращает количество сообщений в почтовом ящике плюс размер, занимаемыми этими сообщениями на почтовом ящике
TOP [N сообщения] [количество строк]	Сервер возвращает заголовки указанного сообщения, пустую строку и указанное количество первых строк тела сообщения.
QUIT	Завершение сессии.

На текущий момент .Net Framework не содержит классов для работы с POP3-сервером.

4. Использование FTP

4.1. 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.

4.2. 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). Удаленная директория — каталог файлов, находящийся на сервере. Удаленный файл — файл на сервере. Локальный файл — файл в файловой системе клиента.

Таблица 9. Некоторые FTP-комманды:

Формат команды	Описание
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.

Формат команды	Описание
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.

Формат команды	Описание
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.

4.3. 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.
```

4.4. Классы для работы с FTP

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

FtpWebRequest — класс для связи с FTP-сервером. При помощи методов этого класса инициируется передача FTP-команд на сервер. Таблица 8. Основные свойства и методы класса FtpWebRequest

Таблица 10. Основные свойства и методы класса FtpWebRequest

№	Метод	Описание
1	WebRequest.Create	Статический метод для создания объекта FtpWebRequest
2	GetResponce	Получает ответ FTP-сервера в виде FtpWebResponce
3	GetRequestStream	Получает поток для выгрузки данных на сервер (upload)
4	BeginGetResponce	Начало асинхронной передачи запроса серверу и получения ответа
5	EndGetResponce	Завершение асинхронной операции получения ответа сервера
6	BeginGet RequestStream	Асинхронное открытие потока для выгрузки на сервер

№	Метод	Описание
7	EndGet RequestStream	Завершение асинхронной операции получения потока для выгрузки на сервер
8	Свойство	Описание
9	Method	Возвращает или задает FTP-команду
10	TimeOut	Интервал ожидания синхронной операции в миллисекундах
11	UsePassive	Пассивный или активный режим передачи
12	UseBinary	Тип данных при передаче
13	RenameTo	Имя переименовываемого файла
14	ReadWriteTimeOut	Время ожидания операции чтения или записи
15	Proxy	Определение прокси-сервера
16	EnableSsl	Использование шифрования при передаче
17	ContentType	Тип содержимого
18	Credentials	Данные для связи с сервером
19	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.

Таблица 11. Основные свойства и методы класса FtpWebResponse

№	Методы	Назначение
1	GetResponse Stream	Получает поток для чтения файла или выгрузки на сервер
2	Close	Освобождает ресурсы
3	Свойства	Описание

№	Методы	Назначение
4	ContentType	Тип содержимого
5	ContentLength	Длина файла
6	BannerMessage	Предварительное сообщение сервера
7	ExitMessage	Сообщение по завершении FTP-сеанса
8	WelcomeMessage	Сообщение сервера после подключения
9	LastModified	Дата последнего изменения файла
10	StatusCode	Текущее состояние FTP-сессии
11	StatusDescription	Описание текущего состояния FTP-сессии
12	IsFromCache	Был ли этот ответ получен из кэша

4.5. 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();
```

5. Экзаменационные задания

Варианты экзаменационных домашних 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.