

Лабораторная работа №5

Отправка почты. Протокол SMTP

Цель работы научиться взаимодействовать с SMTP-сервером для отправки электронных почтовых сообщений, написать простейшую программу на языке C# для отправки сообщений.

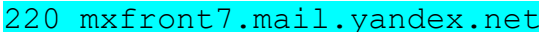
1. Общие сведения о протоколе SMTP

SMTP (Simple Mail Transfer Protocol – "простой" протокол передачи почтовых сообщений) – это протокол прикладного уровня, разработанный в 1982 году для передачи электронных почтовых сообщений в сетях TCP/IP. Протокол SMTP используется для передачи почтового сообщения от отправителя SMTP-серверу, а также для передачи почтовых сообщений между SMTP-серверами. Для получения почтовых сообщений клиентские приложения используют другие протоколы: POP (Post Office Protocol) или IMAP (Internet Message Access Protocol). Работа по протоколу SMTP происходит по принципу клиент/сервер: программа-клиент устанавливает TCP-соединение с сервером (стандартный номер порта – 25/587 (для незащищенного соединения) или 465 (для защищенного соединения) и отправляет ему SMTP-команды в текстовом виде. Сервер обрабатывает команды и выдает отправителю SMTP-ответы. Взаимодействие продолжается, пока клиент или сервер не разорвет TCP-соединение.

1.1 Пример взаимодействия по протоколу SMTP (не защищенное соединение)

 – SMTP-команды, отправляемые клиентом;

 – SMTP-ответы, отправляемые сервером.

 220 mxfront7.mail.yandex.net

Сразу после установления TCP-подключения между клиентом и сервером последний отправляет SMTP-ответ с кодом 220 и своим описанием. Ответы сервера начинаются с кода ответа: 2xx – положительный ответ, 3xx – ожидаются дополнительные данные, 4xx – временный отрицательный ответ, 5xx – постоянный отрицательный ответ.

 HELO sender.example.com

Клиент «приветствует» сервер и сообщает собственное доменное имя.

```
250 mxfront7.mail.yandex.net
```

```
MAIL FROM: <sender@example.com>
```

Команда MAIL FROM устанавливает обратный электронный почтовый адрес.

```
250 2.1.0 <sender@example.com> ok
```

```
RCPT TO: <user@yandex.ru>
```

Команда RCPT TO (Recipient – получатель) устанавливает электронный почтовый адрес получателя.

```
250 2.1.5 <user@yandex.ru> recipient ok
```

```
RCPT TO: <user2@yandex.ru>
```

Если получателей на данном SMTP-сервере несколько, то команда RCPT TO повторяется для каждого из них.

```
250 2.1.5 <user2@yandex.ru> recipient ok
```

```
DATA
```

Командой DATA клиент сообщает, что готов к отправке содержимого почтового сообщения.

```
354 Enter mail, end with "." on a line by itself
```

Сервер предлагает ввести содержимое почтового сообщения. Признаком окончания почтового сообщения является символ «.» на отдельной строке. Почтовое сообщение состоит из заголовка и тела, разделенных пустой строкой. Заголовок не является обязательным. В заголовке указываются поля в формате «ИМЯ_ПОЛЯ: ЗНАЧЕНИЕ_ПОЛЯ». Примеры полей заголовка: To – адрес получателя; From – адрес отправителя; Subject – тема сообщения.

```
To: user@yandex.ru
```

```
From: sender@example.com
```

```
Subject: example of SMTP-message
```

```
text
```

```
text
```

```
text
```

```
.
```

```
250 2.0.0 Ok: queued on mxfront7.mail.yandex.net as
n9uuRiJphy-fOnW1C1
```

Сервер сообщил, что сообщение успешно поставлено в очередь для доставки адресату, а также сообщил ID-номер, присвоенный принятому сообщению. Впоследствии этот номер может быть использован для поиска сообщения в случае, если оно не будет доставлено получателю.

QUIT

Команда для завершения TCP-подключения.

```
221 2.0.0 Closing connection.
```

1.2 Пример взаимодействия по протоколу SMTP (защищенное соединение)

--- – SMTP-команды, отправляемые клиентом;

---- – SMTP-ответы, отправляемые сервером.

```
openssl s_client -crlf -connect smtp.googlemail.com:465
```

Так используется защищенное соединение, то используется SSL-сертификат. SSL-сертификат – это уникальная цифровая подпись, необходимая для организации защищенного соединения между клиентом и сервером, что особенно важно при передаче конфиденциальной информации и проведении финансовых операций. После удачного соединения Вы получите следующие сообщения.

```
CONNECTED(00000003)
```

```
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify error:num=20:unable to get local issuer certificate verify
return:0
```

```
Certificate chain
```

```
0 s:/C=US/ST=California/L=Mountain View/O=Google
Inc/CN=smtp.googlemail.com
i:/C=US/O=Google Inc/CN=Google Internet Authority G2
1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
3 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
```

Server certificate

-----BEGIN CERTIFICATE-----

MIIEijCCA3KgAwIBAgIIWArOc880+NowDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
BhMCVVMxEzARBgNVBAoTCKdvb2dsZSBjbmMxJTAjBgNVBAMTHEdvb2dsZSBjbnRl
cm5ldCBBDxRob3JpdHkgRzIwHhcNMTUxMDEzMjMzMjExWhcNMTYxMDEyMDAwMDAw
WjBtMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcn5pYTEwMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2x1IEluYzEcMBoGA1UEAwwTc210
cC5nb29nbGVtYWlsLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
ALYotACaHYLZGtw+asO6vCXtLz+kjIDQtOLLiGU+p/ykoe5AbB3InEin+KVg+IN7
6ZMSSuR9rtYilodoFERSHsO0MahjB6XujzDGBEY6YrqQ7jEkxbm+uWcbd4xdE9Sh
FQZC9zDRjWM04M8DgcVtKKv+5wcx8hG2NOP6K3JkZT2SJgv4CrIsLERekh1wcJSF
0uhtft7VGulvvDP4/pLNe0BsJ2mZcoommLMLflghUOjGI6am3zYvOK6AoI/89LJs
jPFat/oKazQyRSJS0rm+daWAMjvlp6w4PDrttTnOEWikUGmdHTlknFwFwvzYCSF4
jiX8bC1lFJ3V4d14TKevEKECAwEAAaOCAVAwggFMMB0GA1UdJQQWMBQGCCsGAQUF
BwMBBggrBgEFBQcDAjAeBgNVHREEFzAVghNzbXRwLmdvb2dsZW1haWwuY29tMGgG
CCsGAQUFBwEBBFwwWjArBggrBgEFBQcwAoYfaHR0cDovL3BraS5nb29nbGUuY29t
L0dJQUcyLmNydDARBggrBgEFBQcwAYYfaHR0cDovL2NsaWVudHMxLmdvb2dsZS5j
b20vb2NzcDADBgNVHQ4EFgQUAbd/RAZK+zUSg1C2aOQGeNUEQOEwDAYDVR0TAAQH/
BAIwADAFBgNVHSMEGDAWgBRK3QYWG7z2aLV29YG2u2IaulqBLzAhBgNVHSAEGjAY
MAwGCisGAQQB1nkCBQEwCAYGZ4EMAQICMDAGA1UdHwQpMCcwJaAjoCGGH2h0dHA6
Ly9wa2kuZ29vZ2x1LmNvbS9HSUFHMi5jcmwwDQYJKoZIhvcNAQELBQADggEBABQr
DVnggIUgJDI3a0vAymgPZyHrjO2Ja0/hWT5cHGI fMRxzZJGNP+MadyT8nHcPkNAO
xbJ0OSvkkqFoZqcN64ff4znm7OGTGUI9hS0Exs1xsP+OBkMZ0U4MqilcZzTM426I
N5JUO7NVqxX/VVBSRq7DPPL5koHCV2c0P18Om4Cq2YNXE9nGzWDK5Q11cp6dhPC6
nMmvUrRWBa94b5Wpj/sQld3Xv7wIsYRtQfLCy5sW5OeFSB116+ZwP8PwvUL/pmIn
WuYNIGMijZAxepQV8pdjNtskjGF4mvN9jZ9kl+OgG923/wOA96DBI/2bb2JS4aza
rmXzq0SteCwceGMIjMI=

-----END CERTIFICATE-----

subject=/C=US/ST=California/L=Mountain View/O=Google
Inc/CN=smtp.googlemail.com issuer=/C=US/O=Google Inc/CN=Google
Internet Authority G2

No client certificate CA names sent

SSL handshake has read 3737 bytes and written 421 bytes

New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256

Server public key is 2048 bit

Secure Renegotiation IS supported

Compression: NONE

Expansion: NONE

SSL-Session:

Protocol : TLSv1.2

Cipher : ECDHE-RSA-AES128-GCM-SHA256

Session-ID:

F2C4DE34FEF3CC571B0EA7C569857043F6682DD1606CFCF10850CBED58067768

```

Session-ID-ctx:
Master-Key:
AEC5C5FBF60845197B085F8625CEDD55955D896E2D8FA255530BF53486C33F07
7490FCB913E123D2 C7EA594711010EBA
Key-Arg : None
PSK identity: None
PSK identity hint: None
SRP username: None
TLS session ticket lifetime hint: 100800 (seconds)
TLS session ticket:
0000 - 29 b0 c8 73 ea 29 99 ff-1a d3 88 8d 8c 2c 64 5f
) ..s.) .....,d_
0010 - 43 bd 5e 34 7a df 5d 26-f4 88 3e b2 da d0 d8 14
C.^4z.]&..>.....
0020 - 47 5c e1 3b 55 b3 a7 d0-c4 eb fc 6a 31 66 5a 7d
G\.;U.....j1fZ}
0030 - 20 b6 8c 11 be 54 1f 24-21 8b 7c 15 e1 5a aa 27
....T.$!..|..Z.'
0040 - de ef 37 e9 f7 6d f4 d1-4f c0 31 d7 d4 98 cd 7d
..7..m..O.1....}
0050 - 50 9c 98 76 9a 0d 0e ec-06 3d 49 4a e8 fd 6c 0e
P..v.....=IJ..l.
0060 - 96 41 4b 8c 3c de 8e d1-75 ee be 32 61 d2 1f cf .AK.
0070 - 63 43 07 79 83 72 f9 46-77 37 e9 42 d5 f4 d8 db
cC.y.r.Fw7.B....
0080 - 33 86 02 83 8f b0 bd ae-19 ed 81 e2 35 5a e4 b1
3.....5Z..
0090 - 24 2d 4a 48 34 a4 23 47-03 73 6e ca 9c e7 65 bf $-
JH4.#G.sn...e.
00a0 - 43 85 37 b4 C.7.

Start Time: 1447770580
Timeout : 300 (sec)
Verify return code: 20 (unable to get local issuer
certificate)

```

Далее, как и при не защищенном соединении сервер любезно сообщит Вам код состояния подключения и свое доменное имя.

```
---
```

```
220 smtp.googlemail.com ESMTP ki2sm16061241bc.15 - gsmtip
```

После чего, клиенту необходимо «поприветствовать» сервер командой:

```
EHLO sender.example.com
```

Общение между клиентом и сервером осуществляется в кодировке base64 поэтому по требованию сервера необходимо передать username и password в кодировке base64 (закодировать логин и пароль можно через сайта <http://base64.ru>

```
250-smtp.googlemail.com at your service, [109.227.209.77]
250-SIZE 35882577
250-8BITMIME
#Тут сервер Вам сообщает способы аутентификации
250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER
XOAUTH
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
```

Первый способ (AUTH LOGIN) аутентификации с использованием имени пользователя и пароля отдельными сообщениями.

```
AUTH LOGIN #указываем способ аутентификации
334 VXNlcm5hbWU6 #если перевести с base64 в текст
(получится: Username)
BASE64USERNAME (EMAIL)
334 UGFzc3dvcmQ6 #если перевести с base64 в текст
(получится: Password)
BASE64PASSWORD
235 2.7.0 Accepted
```

Далее по аналогии с незащищённым соединением указываем отправителя, получателя и данные.

2. Взаимодействие с SMTP-сервером в .NET

Для отправки почты в среде интернет используется протокол **SMTP** (Simple Mail Transfer Protocol). Данный протокол указывает, как почтовые сервера взаимодействуют при передаче электронной почты.

Для работы с протоколом SMTP и отправки электронной почты в .NET предназначен класс *SmtplibClient* из пространства имен *System.Net.Mail*.

Этот класс определяет ряд свойств, которые позволяют настроить отправку:

- **Host:** smtp-сервер, с которого производится отправление почты. Например, smtp.yandex.ru

- **Port:** порт, используемый smtp-сервером. Если не указан, то по умолчанию используется 25 порт.

- **Credentials:** аутентификационные данные отправителя

- **EnableSsl:** указывает, будет ли использоваться протокол SSL при отправке

Еще одним ключевым классом, который используется при отправке, является *MailMessage*. Данный класс представляет собой отправляемое сообщение. Среди его свойств можно выделить следующие:

- **Attachments:** содержит все прикрепления к письму

- **Body:** непосредственно текст письма

- **From:** адрес отправителя. Представляет объект *MailAddress*

- **To:** адрес получателя. Также представляет объект *MailAddress*

- **Subject:** определяет тему письма

- **IsBodyHtml:** указывает, представляет ли письмо содержимое с кодом html

Используем эти классы и выполним отправку письма:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;

namespace SMTPDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            // отправитель - устанавливаем адрес и отображаемое в письме имя
            MailAddress from = new MailAddress("dgprivezencev@gmail.com", "Den");
            // кому отправляем
            MailAddress to = new MailAddress("dgprivezencev@mail.ru");
            // создаем объект сообщения
            MailMessage m = new MailMessage(from, to);
            // тема письма
            m.Subject = "Тест";
            // текст письма
            m.Body = "<h2>Письмо-тест работы smtp-клиента</h2>";
            // письмо представляет код html
            m.IsBodyHtml = true;
            // адрес smtp-сервера и порт, с которого будем отправлять письмо
            SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
            // логин и пароль
            smtp.Credentials = new NetworkCredential("dgprivezencev@gmail.com", "*****");
            smtp.EnableSsl = true;
```

```

        smtp.Send(m);
        Console.Read();
    }
}

```

Для отправки применяется метод **Send()**, в который передается объект *MailMessage*.

Также мы можем использовать асинхронную версию отправки с помощью метода *SendMailAsync*:

```

using System;
using System.Net;
using System.IO;
using System.Threading.Tasks;
using System.Net.Mail;

namespace NetConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            SendEmailAsync().GetAwaiter();
            Console.Read();
        }

        private static async Task SendEmailAsync()
        {
            MailAddress from = new MailAddress("dgprivezencev@gmail.com", "Tom");
            MailAddress to = new MailAddress("dgprivezencev@yandex.ru");
            MailMessage m = new MailMessage(from, to);
            m.Subject = "Тест";
            m.Body = "Письмо-тест 2 работы smtp-клиента";
            SmtplibClient smtp = new SmtplibClient("smtp.gmail.com", 587);
            smtp.Credentials = new NetworkCredential("dgprivezencev@gmail.com",
"*****");
            smtp.EnableSsl = true;
            await smtp.SendMailAsync(m);
            Console.WriteLine("Письмо отправлено");
        }
    }
}

```

Добавление вложений

К письму мы можем прикрепить вложения с помощью свойства *Attachments*. Каждое вложение представляет объект *System.Net.Mail.Attachment*:

```

MailAddress from = new MailAddress("dgprivezencev@gmail.com", "Tom");
MailAddress to = new MailAddress("dgprivezencev@yandex.ru");
MailMessage m = new MailMessage(from, to);

```



```
m.Attachments.Add(new Attachment("D://temlog.txt"));
```

Задание на лабораторную работу

1. Выполнить отправку сообщения с использованием программы OpenSSL (Telnet если сервер не требует шифрованного соединения SSL).
2. Доработать клиент-серверное приложение, написанное в лабораторной работе №4. Добавить прием адреса электронной почты от клиента и отправка результата вычислений на полученную почту. Предусмотреть случай, когда клиент пришлет несколько адресов, в этом случае отправить письма не все адреса.