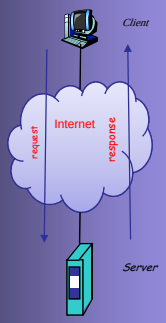


INTERNET PROTOCOLS AND CLIENT-SERVER PROGRAMMING SWE344

Fall Semester 2008-2009 (081)

Module 10.2: Mail Protocols (Part 2)

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives

- Learn about the .NET classes for Constructing and Sending Mail Messages using SMTP protocol
  - MailAddress
  - MailMessage
  - Attachment
  - SmtpClient
- Learn how to write a Mail Sending Application

Introduction

- .NET 1.1 (Visual Studio 2003) provides some classes for constructing and sending mail messages in the System.Web.Mail namespace.
  - The classes were not complete and were dependent on the SMTP server provided by Microsoft Internet Information Server (IIS).
  - For example, the System.Web.Mail.SmtpClient does not work with SMTP servers other than that provided by IIS.
- The new .NET 2.0 provides a much richer set of classes, covering many aspects of SMTP mails.
  - The new set of classes are in a new namespace, System.Net.Mail.
- Some of the classes in the System.Net.Mail are:
  - MailAddress
  - MailMessage
  - SmtpClient
  - Attachment

The MailAddress class

- The MailAddress class represents an e-mail address.
- Constructors:

```
MailAddress (string address)
MailAddress (string address, string displayName)
```

- Properties:

string Address	Gets the e-mail address
string DisplayName	Gets the display name
string Host	Gets the host portion of the address
string User	Gets the username portion of the address

- Note: There is also the MailAddressCollection class that stores a collection of MailAddress objects.

The MailMessage Class

- The MailMessage class is used to represent an e-mail message (in the RFC 2822 format) as an object.
- Constructors:

```
MailMessage ()
MailMessage (MailAddress from, MailAddress to)
MailMessage (string from, string to)
MailMessage (string from, string to, string subject, string body)
```

- Properties:

MailAddress From	Gets or sets the from address
MailAddressCollection To	Gets the address collection that contains the recipients. Addresses are added using the Add method.
MailAddressCollection CC	Gets the address collection that contains the carbon copy (CC) recipients.

The MailMessage Class …

- More Properties:

MailAddressCollection Bcc	Gets the address collection that contains the blind carbon copy (BCC) recipients.
string Subject	Gets or sets the message subject line.
string Body	Gets or sets the message body.
AttachmentCollection Attachments	Gets the attachment collection. Use the Add method to add Attachment objects.
Encoding BodyEncoding	Gets or sets the encoding used to encode the message body. Default is "us-ascii" .
NameValueCollection Headers	Gets the e-mail headers. Use Add method to add more headers.
bool IsBodyHtml	Gets or sets a value indicating whether the mail message body is in Html.

The SmtpClient Class

- The SmtpClient class allows applications to send e-mail message to an SMTP server for delivery. It hides the details of the SMTP commands and replies from the user.
- To send an e-mail message using SmtpClient, the following are required:
  - The SMTP host server that will be used to send the message.
  - Credentials for authentication, if required by the SMTP server.
  - The e-mail address of the sender
  - The e-mail address or addresses of the recipients.
  - The message content.
- Constructors:

```
SmtpClient ()
SmtpClient (string host)
SmtpClient (string host, int port)
```


The SmtpClient Class …

- Properties:

ICredentialsByHost Credentials	Gets or sets the credentials used to authenticate the sender.
SmtpDeliveryMethod DeliveryMethod	Specifies how outgoing email messages will be handled. This is an enumeration with values: ▪**Network**: Email is sent through the network to an SMTP server. ▪**PickupDirectoryFromIis**: Email is copied to the pickup directory used by a local IIS for delivery. ▪**SpecifiedPickupDirectory:** Email is copied to the directory specified by the SmtpClient.PickupDirectoryLocation property
string PickupDirectoryLocation	Gets/sets the folder that messages are saved for processing by the local SMTP server.
string Host	Gets/sets the name or IP address of the host
int Port	Gets or sets the port

The SmtpClient Class …

- More Properties:

int Port	Gets or sets the port
int Timeout	Gets/sets a value that specifies the amount of time after which a synchronous Send call times out. The default value is 100,000 (100 seconds)
bool UseDefaultCredentials	Gets/sets a Boolean value that controls whether the DefaultCredentials are sent with requests.

The SmtpClient Class …

- Methods:

Send (MailMessage message) Send (string from, string to, string subject, string body)	Sends the message to an SMTP server for delivery.
SendAsync (MailMessage message, object state) SendAsync (string from, string to, string subject, string body, object state)	Sends the message to the SMTP server asynchronously.
SendAsyncCancel()	Cancels an asynchronous operation to send an e-mail message

- Events:

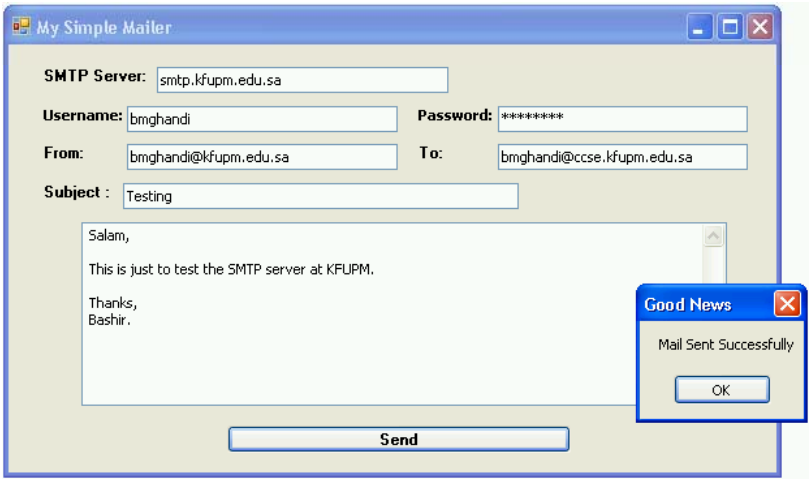
SendCompletedEventHandler SendCompleted	Fired when the operation to send the message using asynchronous method is complete

Note: The signature of the SendCompletedEventHandler delegate is:

void SendCompletedCallback (object sender, AsyncCompletedEventArgs e)

Example 1

- The following example is a simple mail sending application.



Example 1 …

```
1.  void btSendClick(object sender, System.EventArgs e) {
2.    MailMessage mail = new MailMessage();
3.    mail.From = new MailAddress(txtFrom.Text);
4.    mail.To.Add(new MailAddress(txtTo.Text));
5.    mail.Subject = txtSubject.Text;
6.    mail.Body = txtBody.Text;
7.    //Or simply
8.    //MailMessage mail = new MailMessage(txtFrom.Text,
9.    //        txtTo.Text, txtSubject.Text, txtBody.Text);

10.   SmtpClient client = new SmtpClient(txtServer.Text);
11.    client.Credentials = new NetworkCredential(txtUsername.Text,
12.                                txtPassword.Text);
13.    //client.Send(mail);
14.    // or better to use asynchronous method as follows

15.     client.SendCompleted +=
16.           new SendCompletedEventHandler(SendCompletedCallback);
17.     client.SendAsync(mail, null);
18. }
```

Example 1 …

```
1.  public static void SendCompletedCallback(object sender
2.                              AsyncCompletedEventArgs e) {
3.    if (e.Cancelled)
4.       MessageBox.Show("Send Operation Cancelled",
5.                                 "Error Sending Message");
6.    else if (e.Error != null)
7.       MessageBox.Show(e.Error.ToString(),
8.                                 "Error Sending Message");
9.    else
10.      MessageBox.Show("Mail Sent Successfully",
11.                                       "Good News");
12.   }

13. public static void Main(string[] args)
14. {
15.   Application.Run(new SimpleMailer());
16. }
17.
```


The Attachment Class

- The Mail Format Protocol (RFC 2822) does not directly support sending Mail Attachments – body must be in US-ASCII
- The MIME protocol (discussed in the next lecture) introduced a mechanism for adding attachments to RFC 2822 mails without violating the protocol:
  - Attachments are encoded into US-ASCII using an encoding scheme such as Base64, and appended to the mail body for transportation.
  - Various headers are used to indicate the content-type, transfer encoding method and boundaries of each attachment so that it can be decoded and retrieved correctly at the receiving end.
  - The Attachment class provides various properties that are used to set the transfer-encoding method and the relevant MIME headers.
  - Instances of the Attachment class are added to the mail body using the Attachments property of the MailMessage instance.
  - Attachments is of type AttachementCollection, which is another class in the System.Net.Mail namespace.

The Attachment Class …

- Constructors:

```
Attachment (string path)
Attachment (Stream contentStream, ContentType contentType)
Attachment (Stream contentStream, string contentTypeName)
Attachment (string path, ContentType contentType)
Attachment (string path, string mediaType)
Attachment (Stream contentStream, string name, string mediaType)
```

- Properties:

string ContentId	Gets/sets the MIME content ID for the attachment.
Stream ContentStream	Gets the content stream of the attachment
ContentType ContentType	Gets the content type of this attachment. ContentType has various properties such as: Boundary, MediaType, Name, CharSet, etc.

The Attachment Class …

string Name	Gets/sets the MIME content type name value in the content type of the attachment
TransferEncoding TransferEncoding	Gets or sets the encoding of the attachment. TrasferEncoding is a enum consisting of: ▪ **Base64** : For encoding binary data (default for attachments) ▪ **QuotedPrintable** : For encoding data that consists of printable characters (default for main body). ▪ **SevenBit** : For data that is not encoded
ContentDisposition ContentDisposition	Gets the MIME content disposition for the attachment. ContentDisposition is used to specify how the receiver should handle the attachment. ▪ It has a bool property, Inline, which if true means the attachment should be displayed with the mail. ▪ Other properties include: FileName, Size, etc.

The Attachment Class …

Notes:

- When creating an Attachment instance, If content type is not specified, the default application/octet-stream is assumed.
- Also If contentName is not provided, the name is obtained from the path.
- The following classes are provided in the System.Net.Mime namespace to represents the three main media Types.
  - The public fields of these classes can be used to specify MIME MediaTypes.

Class	Public Fields
MediaTypeNames.Application	Octet, Pdf, Rtf, Soap, Zip
MediaTypeNames.Image	Gif, Jpeg, Tiff
MediaTypeNames.Text	Html, Plain, RichText, Xml

Example 2: Sending Attachments

```
1.  public static void CreateMessageWithAttachment(
2.                                          string server) {
3.      string file = "lab03.pdf";
4.      MailMessage message = new MailMessage(txtFrom.Text,
5.                txtTo.Text, txtSubject.Text, txtBody.Text);
6.      // Create and add the attachment.
7.      Attachment data = new Attachment(file,
8.                            MediaTypeNames.Application.Pdf);
9.      message.Attachments.Add(data);
10.     //Send the message.
11.     SmtpClient client = new SmtpClient(server);
12.     // Add credentials.
13.     client.Credentials = new NetworkCredential(
14.                     txtUsername.Text, txtPassword.Text);
15.     client.Send(message);
16. }
```

Resources

- MSDN Library
  - http://msdn.microsoft.com/en-us/default.aspx
- Books
  - Richard Blum, C# Network Programming. Sybex 2002.
- Lecture notes of previous offerings of SWE344 and ICS343
- Some other web sites and books; check the course website at
  - http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/swe344/index.htm