

Envío de e-mails vía SMTP con GTK# y Monodevelop.

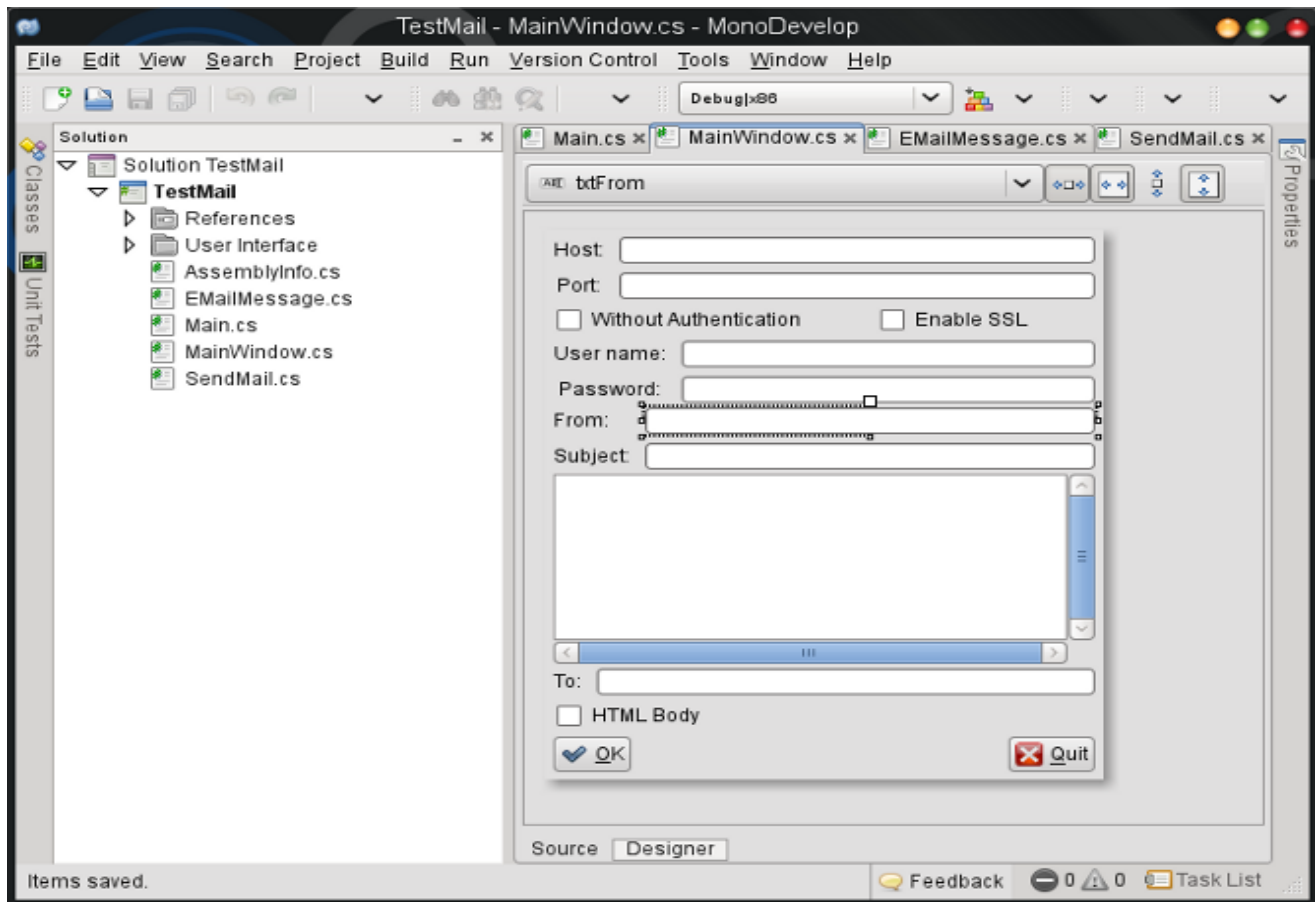
por Martín A. O Márquez <xomalli@gmail.com>

Uno de los requerimientos más comunes para las aplicaciones .NET es el envío de E-mails o correos electrónicos, entre los ejemplos mas usuales para este tipo de requerimiento se encuentran el envío automático de boletines electrónicos, notificaciones de eventos, bitácora del sistema, solicitudes de reunión, envío de archivos adjuntos entre otros. .NET nos proporciona los ensamblados System.Net.Mail y System.Net.Mime, los cuales contienen todas las clases e interfaces necesarias para habilitar este tipo de requerimiento en cualquier aplicación .NET, a continuación describimos los elementos más importantes de estos ensamblados:

- [MailMessage](#): Representa un mensaje de correo electrónico, esta clase tiene todas las propiedades del mensaje por ejemplo: tema del mensaje, destinatario, destinatarios, remitente, cuerpo entre otras.
- [MailAddress](#): Representa una dirección de correo electrónico, esta clase ya implementa la lógica para validar que el formato sea correcto.
- [ContentType](#): Representa la cabecera del protocolo MIME
- [AlternateView](#): Representa una vista alternativa del mensaje en un formato diferente al predeterminado, por ejemplo HTML ó Calendar.
- [SmtpClient](#): Esta clase se encarga de enviar el mensaje, aquí se establecen los parámetros de configuración del servidor SMTP, como: numero de puerto, usuario, password, SSL entre otros.

Como ejemplo de la utilización de estas clases, mostramos un formulario GTK# en Monodevelop que recibe los parámetros de configuración de un servidor SMTP, crea un mensaje de correo electrónico y lo envía hacia el o los destinatarios, utilizando dos clases: *EMailMessage* y *SendMail*.

Fig1 El formulario en vista de diseño de MonoDevelop.



El código fuente de la clase *EmailMessage*

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

namespace TestMail
{
    public class EmailMessage : MailMessage
    {
        public EmailMessage (string subject, string fromField, List<string> to, string body,
                             bool isBodyHtml)
        {
            Subject = subject;
            From = new MailAddress(fromField);
            foreach (string item in to) To.Add(new MailAddress(item));
            IsBodyHtml = isBodyHtml;
            if (IsBodyHtml)
                Body = GetHTMLContent(body);
            else

```

```

    Body = body;
}

string GetHTMLContent(string body){
return new StringBuilder().Append("<html><head><title>")
.AppendFormat("{0}</title></head>", Subject)
.AppendFormat("<body><p><h2><font face=\"arial\" "
color=\"green\">{0}</font></h2></p><body></html>",
body).ToString();
}
}
}

```

El código fuente de la clase *Sendmail*

```

using System;
using System.Net.Mail;
using System.IO;
using System.Net;
using System.Net.Security;
using System.Security;
using System.Security.Cryptography.X509Certificates;

namespace TestMail
{
    public class SendMail
    {
        public string Server {private set;get;}
        public string UserName {private set;get;}
        string Password {set;get;}
        public int Port {private set;get;}
        public bool EnableSSL {private set;get;}
        bool Credentials;
        SmtplibClient SmtplibClient = null;
        public SendMail(string server,int port,bool enableSSL){
            Server = server;
            Port = port;
            EnableSSL = enableSSL;
        }
        public SendMail(string server,int port,bool enableSSL,string username,string
password):
            this(server,port,enableSSL){
            UserName = username;
            Password = password;
            Credentials = true;
        }

        public void Send(MailMessage msg){
            SmtplibClient = new SmtplibClient(Server,Port);
            SmtplibClient.EnableSsl = EnableSSL;
            //DO NOT use: this property throws an Exception
            //SmtplibClient.UseDefaultCredentials = true;
            if(Credentials){
                SmtplibClient.UseDefaultCredentials = false;
                SmtplibClient.Credentials = new NetworkCredential(UserName,Password);
            }
            ServicePointManager.ServerCertificateValidationCallback =

```

```

    delegate(object s, X509Certificate certificate, X509Chain chain, SslPolicyErrors
sslPolicyErrors)
    { return true; };
    SmtplibClient.Send(msg);
}
}
}

```

El código fuente del formulario GTK#.

```

using System;
using Gtk;
using TestMail;
using System.Collections.Generic;

public partial class MainWindow : Gtk.Window
{
    public MainWindow () : base(Gtk.WindowType.Toplevel)
    {
        Build ();
    }

    protected void OnDeleteEvent (object sender, DeleteEventArgs a)
    {
        Application.Quit ();
        a.RetVal = true;
    }
    protected virtual void OnBtnSubmitClicked (object sender, System.EventArgs e)
    {
        try{
            SendMail sendMail = null;
            var email = new EmailMessage(txtSubject.Text,
                                         txtFrom.Text,
                                         GetAddresses(),
                                         txtDescription.Buffer.Text,
                                         chkIsHtml.Active);
            if(chkCredentials.Active)
                sendMail = new
                SendMail(txtHost.Text, Convert.ToInt32(txtPort.Text), chkEnableSSL.Active);
            else
                sendMail = new
                SendMail(txtHost.Text, Convert.ToInt32(txtPort.Text), chkEnableSSL.Active
                        , txtUserName.Text, txtPassword.Text);
            sendMail.Send(email);
            ShowMessageBox("Email Sent");
        }catch(Exception ex){
            ShowMessageBox(ex.Message);
        }
    }

    void ShowMessageBox(string msg){
        using (Dialog dialog = new MessageDialog (this,
            DialogFlags.Modal | DialogFlags.DestroyWithParent,
            MessageType.Info,
            ButtonsType.Ok,
            msg)) {
            dialog.Run ();
            dialog.Hide ();
        }
    }
}

```

```

}
}

List<string> GetAddresses() {
var resp = new List<string>();
foreach (string s in txtTo.Text.Split(','))
resp.Add(s);
return resp;
}
protected virtual void OnBtnQuitClicked (object sender, System.EventArgs e)
{
Application.Quit();
}

protected virtual void OnChkCredentialsToggled (object sender, System.EventArgs e)
{
tblCredentials.Visible = !chkCredentials.Active;
}
}

```

Básicamente el envío de correos electrónicos consiste de dos pasos: Primero la creación del mensaje con sus adjuntos si es que los hubiese, de esta responsabilidad se encarga la clase *EMailMessage*, la cuál es una clase derivada de *MailMessage* y se establecen todos sus parámetros en el constructor.

```

public EMailMessage (string subject,string fromField,List to,string body,
                    bool isBodyHtml)
{
Subject = subject;
From = new MailAddress(fromField);
foreach(string item in to) To.Add(new MailAddress(item));
IsBodyHtml = isBodyHtml;
if(IsBodyHtml)
Body = GetHTMLContent(body);
else
Body = body;
}

```

Como segundo paso, el envío del mensaje mediante un servidor SMTP. La clase *Sendmail* se ocupa de esta responsabilidad solicitando los parámetros de configuración del Server.

```

public SendMail(string server,int port,bool enableSSL){
Server = server;
Port = port;
EnableSSL = enableSSL;
}
public SendMail(string server,int port,bool enableSSL,string username,string
password):
this(server,port,enableSSL){
UserName = username;
Password = password;
Credentials = true;
}

```

Y después envía el mensaje con el siguiente método:

```
public void Send(MailMessage msg){
    SmtplibClient = new SmtplibClient(Server,Port);
    SmtplibClient.EnableSsl = EnableSSL;
    //DO NOT use: this property throws a Exception
    //SmtplibClient.UseDefaultCredentials = true;
    if(Credentials){
        SmtplibClient.UseDefaultCredentials = false;
        SmtplibClient.Credentials = new NetworkCredential(UserName,Password);
    }
    ServicePointManager.ServerCertificateValidationCallback =
        delegate(object s, X509Certificate certificate, X509Chain chain,
        SslPolicyErrors sslPolicyErrors)
        { return true; };
    SmtplibClient.Send(msg);
}
```

En este método primeramente establecemos la configuración del servidor, indicamos si se usa o no una conexión segura, después en caso de requerir identificación establecemos el usuario y la contraseña, después utilizamos el delegado *ServerCertificateValidationCallback* de la clase estática [ServicePointManager](#) para la validación del certificado del servidor. Luego de compilar, ejecutamos y probamos la aplicación utilizando el servidor de correo *Gmail* en cuyo caso establecemos la configuración como:

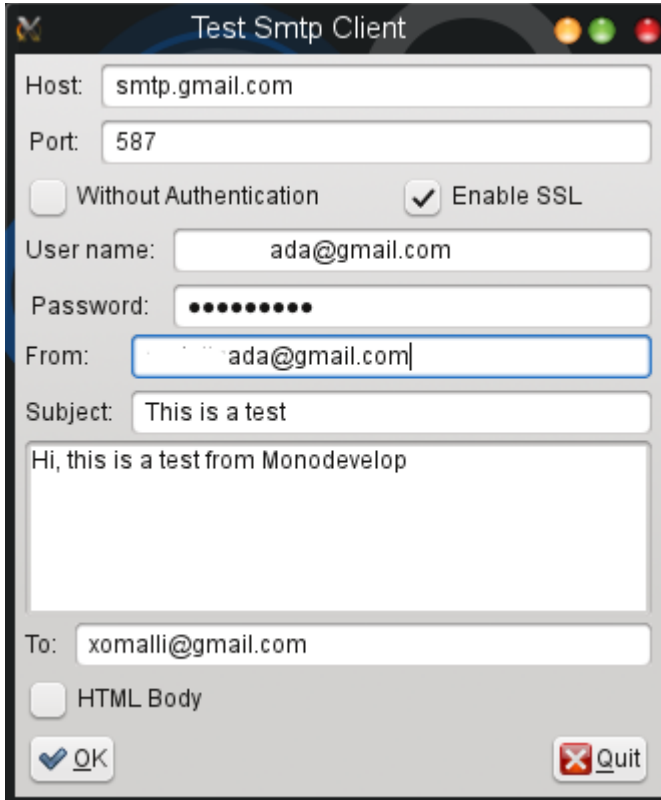
Host: smtp.gmail.com

Port: 587

Username: [user name]@gmail.com

EnableSSL: true

Fig 2 Configurando los parámetros para la salida de mensajes

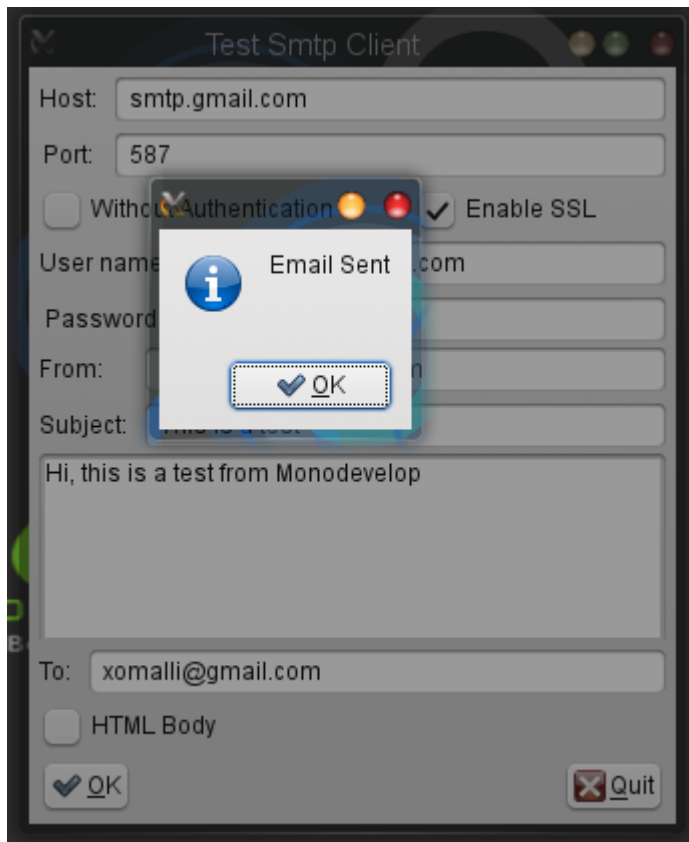


The image shows a window titled "Test Smtplib Client" with the following fields and controls:

- Host: smtp.gmail.com
- Port: 587
- ☐ Without Authentication ☒ Enable SSL
- User name: ada@gmail.com
- Password: (masked with dots)
- From: ada@gmail.com
- Subject: This is a test
- Body text: Hi, this is a test from Monodevelop
- To: xomalli@gmail.com
- ☐ HTML Body
- Buttons: OK (with a checkmark icon) and Quit (with a red X icon)

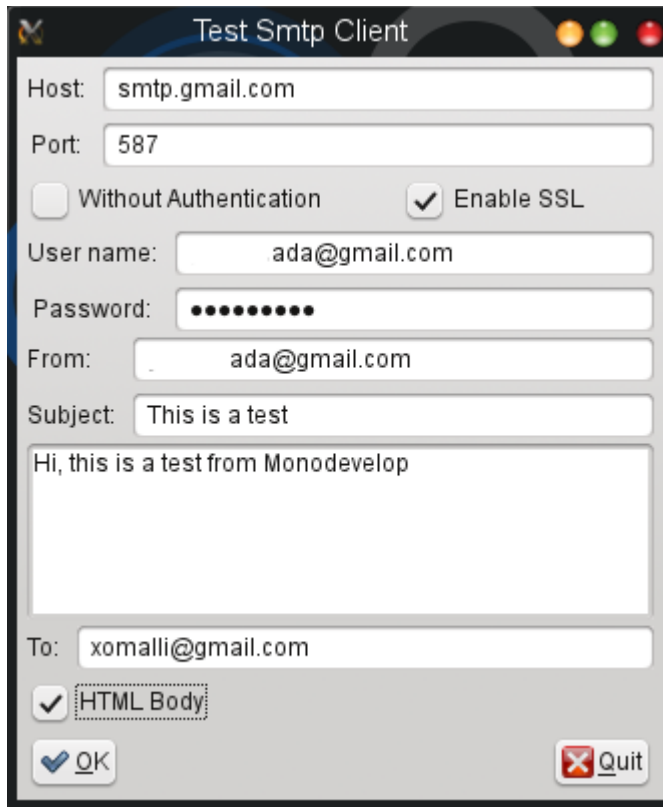
Si el envío resulta exitoso la aplicación nos mostrará el mensaje como en la siguiente figura.

Fig 3 Enviando el mensaje con éxito



De forma predeterminada el mensaje se envia en texto plano, para enviar el mensaje en HTML, habilitamos el checkbox para activar la propiedad *IsBodyHtml* y recibir el mensaje en este formato.

Fig 4 Habilitando soporte a HTML en el cuerpo del mensaje



The image shows a window titled "Test Smt Client" with a standard Linux-style title bar (yellow, green, and red buttons). The window contains several input fields and checkboxes for configuring an SMTP client. The "Host" field is set to "smtp.gmail.com" and the "Port" field is set to "587". There are two checkboxes: "Without Authentication" (unchecked) and "Enable SSL" (checked). The "User name:" field contains ".ada@gmail.com" and the "Password:" field is masked with dots. The "From:" field also contains ".ada@gmail.com" and the "Subject:" field contains "This is a test". Below these is a large text area containing the message body: "Hi, this is a test from Monodevelop". The "To:" field is set to "xomalli@gmail.com". At the bottom, there is a checkbox for "HTML Body" which is checked, and two buttons: "OK" and "Quit".

Host: smtp.gmail.com

Port: 587

☐ Without Authentication ☒ Enable SSL

User name: .ada@gmail.com

Password:

From: .ada@gmail.com

Subject: This is a test

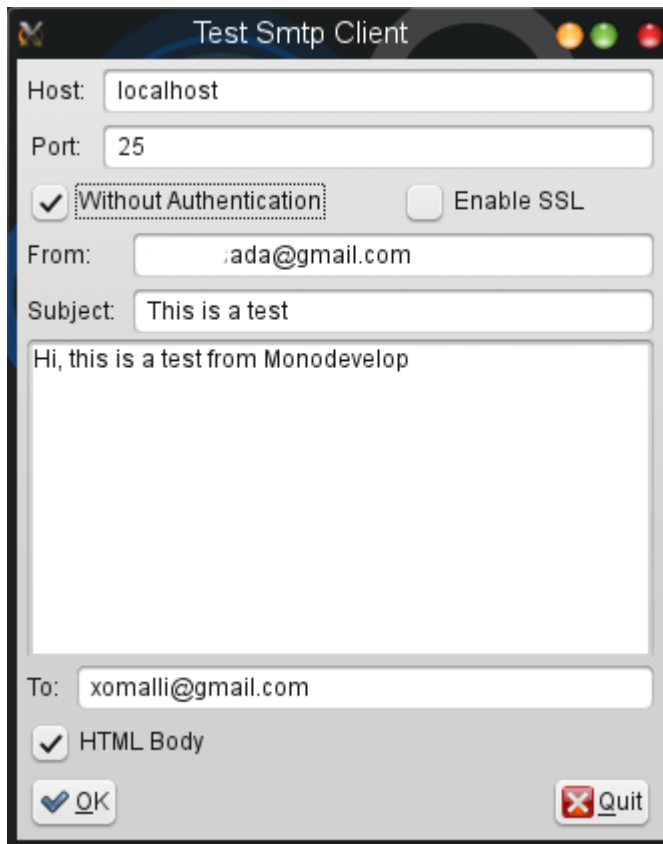
Hi, this is a test from Monodevelop

To: xomalli@gmail.com

☒ HTML Body

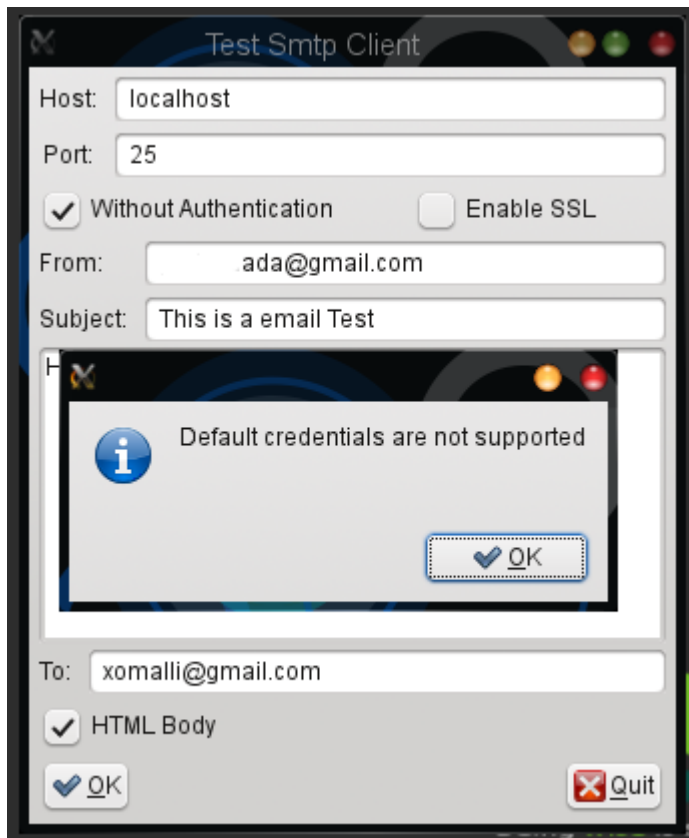
Tambien podemos utilizar un servidor local SMTP en caso de que nuestra distribución [OpenSuse](#) tenga instalados *Postfix* o *Sendmail*.

Fig 5 Utilizando un servidor SMTP local.



En el caso de no usar autenticación existe la propiedad *UseDefaultCredentials* **que funciona en el Framework .NET de Microsoft, no así en el Framework Mono**, si la utilizamos, Mono lanzará una excepción como se muestra en la siguiente imagen:

Fig 6 Mono no incluye el soporte para la propiedad *UseDefaultCredentials*



[Descarga el código fuente en un proyecto para MonoDevelop o Visual Studio](#)

Este documento está protegido bajo la licencia de documentación libre *Free Documentacion License* del Proyecto GNU, para consulta ver el sitio <http://www.gnu.org/licenses/fdl.txt> , toda persona que lo desee está autorizada a usar, copiar y modificar este documento según los puntos establecidos en la «Licencia FDL»