Michaël Van Canneyt,
Noud van Kruysbergen

Mailen met Lazarus

Programmeercursus Pascal, deel 4: e-mailen

Er bestaan verschillende componenten om allerlei TCP/IP-protocollen af te handelen met Lazarus. Het versturen van e-mail via het SMTP-protocol is daar een voorbeeld van. Een bijzonder makkelijke manier om e-mails te verzenden is Synapse.

Soms wil je vanuit een programma of applicatie een mailtje kunnen sturen, bijvoorbeeld bij een website waarop gebruikers zich kunnen inschrijven voor een toernooi en je dan een bevestigingsmail moet terugsturen. Of een Windows-service die een fout aan de systeembeheerder moet melden of een waarschuwing via e-mail moet geven als de harde schijf vol dreigt te lopen.

In deze aflevering van de Pascal-cursus gaan we in op de mogelijkheden om e-mails te sturen. De oplettende lezer heeft ongetwijfeld gemerkt dat dit eigenlijk het onderwerp van het vijfde deel van de cursus zou zijn, maar dat komt omdat we het oorspronkelijke deel over het gebruik van multimedia binnen Pascal één c't uitstellen, om er dan ook iets heel moois van te maken. Maar daarover laten we je nog even in spanning en volstaan we met de mededeling dat de inhoud van dit deel van de cursus de volgende keer ook handig is... Maar zoals gezegd, gaan we in dit deel eerst kijken naar de netwerkmogelijkheden van Pascal aan de hand van SMTP.

Indy of Synapse

Er bestaan verschillende klassenbibliotheken die mails kunnen versturen. Indy en Synapse zijn er daar twee van, die beide met Delphi en Lazarus gebruikt kunnen worden. Indy is meer component-based en Synapse is een verzameling classes die allerlei TCP/IP-taken afhandelen. Indy is dan ook wat lastiger en complexer in het gebruik, maar biedt daarentegen meer mogelijkheden en stabiliteit voor grotere projecten. Het voordeel van Synapse is bovendien dat het naadloos werkt met Free Pascal en Lazarus, dus vandaar dat we hier voor Synapse hebben gekozen.

Synapse wordt niet standaard met Lazarus meegeleverd, maar bij het nieuwste installatiebestand van Lazarus (zie de softlink aan het eind van dit artikel) zijn zowel Synapse als Indy voorgeïnstalleerd. Je kunt Synapse ook downloaden van http://synapse.ararat.cz/.

Na het installeren van Lazarus via het installatieprogramma op de ISO staat Synapse in de map c:\lazarus\components\synapse40\. Het project zelf staat daar met alle code en als zipfile. In de map /source/lib staat een laz_synapse.lpk package-bestand. Omwille van het gemak hebben we deze componentengroep al voorgeïnstalleerd.

Als je wilt uitgaan van je bestaande Lazarus-installatie, dan staat het pakket ook los op de installatie-cd en kun je het openen met de package-manager van Lazarus. Ga naar 'Package / Open Pakket bestand (.lpk)…' Het volstaat om de package één keer te compileren door te klikken op 'Compileren'. Het hoeft niet geïnstalleerd te worden in de IDE, want Synapse bevat zoals gezegd geen componenten die op de componentenbalk geïnstalleerd worden. Alle klassen moeten in de code aangemaakt en gebruikt worden.

Om Synapse aan je project toe te voegen, klik je op 'Use…' op de werkbalk van de pakketmanager en op 'Voeg toe aan project'. In de Project Inspecteur ('Project / Project Inspecteur') verschijnt Synapse dan als pakket.

Als je wilt, is er wel een apart package 'visualsynapse' met een verzameling componenten die op de componentenbalk geïnstalleerd kunnen worden. Deze package wordt echter niet ondersteund door de maker van Synapse.

Eenmaal gecompileerd, kan de package als dependency van een project gekozen worden.

Mail via SMTP

E-mailprogramma's maken gebruik van het protocol SMTP (Simple Mail Transfer Protocol) om e-mails te versturen. Dat betekent dat ze een e-mail afleveren aan de SMTP-server (een MTA of Mail Transfer Agent) die de verdere aflevering naar de juiste bestemming voor zijn rekening neemt. Meestal gebeurt de communicatie met een SMTP-mailserver via TCP/IP-poort 25.

Het SMTP-protocol is niet zo complex, waardoor het versturen van een mailtje dan ook een eenvoudige aangelegenheid is. In Synapse kan dat met een functie die in de unit smtpsend gedefinieerd is:

```
function SendTo(const MailFrom, MailTo,
    Subject, SMTPHost: string;
    const MailData: TStrings): Boolean;
```

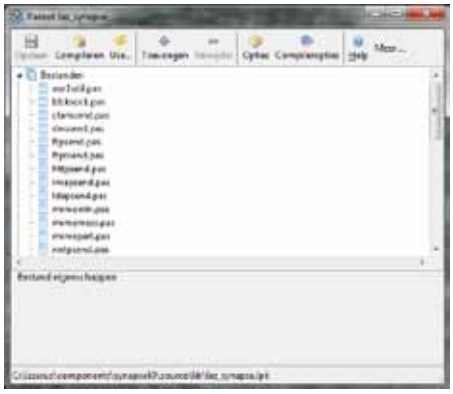
De vijf parameters spreken redelijk voor zich. MailFrom is het e-mailadres van de afzender. MailTo bevat de e-mailadressen van de ontvangers, gescheiden door een komma. Er mogen geen namen in staan, alleen de feitelijke e-mailadressen. Het volgende geeft dus een fout:

```
Michael Van Canneyt <michael@freepascal.org>
```

Maar dit zal wél goed functioneren:

```
michael@freepascal.org
```

Het onderwerp van de e-mail staat als string in



Het Synapse-pakket is makkelijk aan je Lazarus-installatie toe te voegen. Voor het gebruik moet je het eerst eenmaal compileren.

Subject. Dit wordt aan de header van de mail toegevoegd. In SMTPHost komt het IP- of DNS-adres van de SMTP-server die het bericht zal afleveren. Dat is meestal de uitgaande mailserver van je internetprovider.

In MailData komt dan het eigenlijke e-mailbericht te staan.

De functie SendTo maakt een instance aan van de klasse TSMTPSend en vult de nodige properties in, waarna de methode MailTo gebruikt wordt om de mail naar alle opgegeven e-mailadressen te sturen. Als alles correct verlopen is, wordt True teruggegeven.

Sommige mailservers vereisen een combinatie van gebruikersnaam en wachtwoord voor het versturen van e-mails. Daar is de functie SendToEx voor bedoeld, die werkt identiek aan SendTo, maar staat toe een gebruikersnaam en wachtwoord op te geven:

```
function SendToEx(const MailFrom, MailTo,
    Subject, SMTPHost: string;
    const MailData: TStrings;
    const Username, Password: string): Boolean;
```

In feite is de functie SendTo niet meer dan een speciaal geval van de functie SendToEx, waarbij de Username en het Password leeg worden gelaten.

Het pakket laz-synapse is toegevoegd aan het project, zodat de functies binnen je code gebruikt kunnen worden.

Eerste mailprogramma

Daarmee hebben we eigenlijk alles wat nodig is om een eenvoudig mailprogramma in elkaar te knutselen. Het project 'simplesmtp' is via de softlink te downloaden. Pak het zip-bestand uit en kopieer het naar de map met je broncodes en open het projectbestand simplesmtp.lpi.

Er zijn enkele configuratievariabelen nodig voor het e-mailadres van de afzender, de SMTP-server, en de gebruikersnaam en het wachtwoord. Deze zijn als eenvoudige form-variabelen te definiëren:

```
FSender,
FSMTPHost,
FSMTPUser,
FSMTPPasswd : String;
```

Deze variabelen kunnen via een configuratiedialoog ingevuld worden, waarna ze in een INI-bestand bewaard worden en dan opgehaald bij het starten van het programma. Deze code is redelijk eenvoudig, die bekijken we hier dan ook niet in detail. In het broncodebestand frmsettings.pas kun je zien hoe dat gedaan wordt.

Daarna moeten we de mogelijkheid hebben om een e-mailtje samen te stellen. Dat vergt twee TEdit-controls voor de ontvangers en het onderwerp van de e-mail (respectievelijk ETo en ESubject), plus een TMemo (MMail) om de tekst van het e-mailbericht in te voeren.

Dan resteert alleen nog een toolbar met een knop voor het 'Verzenden' van de e-mail en daarmee is de eerste elementaire interface voor het programma al klaar.

De knop 'Verzenden' is aan een actie gekoppeld, waarvan we de OnExecute-eventhandler als volgt instellen:

```
procedure TMainForm.AVerzendenExecute(Sender: TObject);
begin
    if not CheckSettings then
        Exit;
    DoSendMail(ETo.Text,ESubject.Text,MMail.Lines);
end;
```

De functie CheckSettings gaat na of er een afzenderadres en SMTP-serveradres in de form-variabelen ingevuld zijn. Als dat niet het geval is, wordt een boodschap getoond:

```
Function TMainForm.CheckSettings : Boolean;
begin
    Result:=(FSender<>'') and (FSMTPHost<>'');
    if not Result then
                ShowMessage('Geen afzender of mailserver
                                                ingesteld');
end;
```

Als alles goed lijkt te zijn, wordt de functie DoSendMail opgeroepen. Deze functie roept eenvoudigweg de functie SendTo of SendToEx op:

```
procedure TMainForm.DoSendMail(Const ATo,ASubject : String;
    Content : TStrings);
Var
    B : Boolean;
begin
```

Het formulier voor het opstellen en versturen van een e-mail is redelijk overzichtelijk. Er zijn knoppen voor de meest basale functies als het versturen en het toevoegen van bijlages.

```
    if (FSMTPUser<>'') then
        B:=SendToEx(FSender,ATo,ASubject,FSMTPHost,
        Content,FSMTPUser,FSMTPPasswd)
    else
        B:=SendTo(FSender,ATo,ASubject,FSMTPHost,
        Content);
    if not B then
        ShowMessage('Kon het bericht niet verzenden!')
    else
        ShowMessage('Bericht verzonden.');
end;
```

Na een succesvol verloop van de operatie wordt een bevestiging getoond dat de e-mail verzonden is.

Meer dan dit is er in de basis niet nodig om vanuit een programma een e-mail te sturen.

Bijlagen

Soms wil je een HTML-mail sturen, of een bijlage toevoegen met een PDF-bestand of foto. Het SMTP-protocol staat echter alleen het versturen van platte tekst toe. Om bijvoorbeeld een foto als een bijlage te versturen, moet MIME-encodering toegepast worden. MIME staat daarbij voor Multipurpose Internet Mail Extensions. MIME-encodering zorgt ervoor dat een willekeurig aantal bestanden als teksten in het e-mailbericht kunnen worden gezet. Elk e-mailprogramma kan dat en omgekeerd kunnen van binnenkomende e-mails ook de bijlages opnieuw samengesteld worden uit de MIME-mailtekst.

Een MIME-bericht bestaat uit verschillende delen (parts). Die zijn verschillend van

Het formulier voor de standaardconfiguratie. Deze instellingen worden bewaard in een INI-bestand om niet iedere keer ingevoerd te hoeven worden.

Onze simpele e-mailclient ziet er dan uiteindelijk zo uit.

aard. Het MIMI-type Text bevat platte tekst, in een willekeurige karakterset. Het type Binary bevat binaire data. Het type Message bevat een compleet (ander) e-mailbericht. Als laatste bevat het type Multipart een of meerdere andere parts. Om een e-mail op te stellen met een bijlage, moeten we een Multipart-part maken waaraan dan een Text-part (voor de tekst van de e-mail) en een (of meerdere) Binary-parts worden toegevoegd. Het geheel wordt dan als een e-mailbericht verzonden.

Synapse biedt een klasse om het samenstellen van een MIME-mailtekst te vergemakkelijken: TMimeMess. Deze klasse verzamelt enkele parts (een of meerdere instances van TMimePart) en stelt dan de MIME-inhoud van het bericht samen. Deze klasse zorgt ook voor het verzamelen van de headers voor het mailbericht.

Een part wordt toegevoegd door middel van de functie AddPart:

```
function AddPart(const PartParent: TMimePart): TMimePart;
```

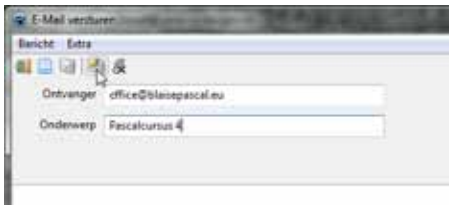
De klasse TMimePart heeft een hele bulk aan properties die de inhoud beschrijven. In principe moeten deze correct ingevuld worden om ervoor te zorgen dat de TMimeMess-instance het MIME-bericht goed kan samenstellen.

Gelukkig zijn er wat hulpmethodes die deze properties automatisch instellen. Je kunt een MultiPart-deel toevoegen via:

```
function AddPartMultipart(const MultipartType: String;
    const PartParent: TMimePart): TMimePart;
```

Tekst kun je dan toevoegen via een van de volgende functies:

```
function AddPartText(const Value: TStrings;
    const PartParent: TMimePart): TMimepart;
function AddPartTextEx(const Value: TStrings;
    const PartParent: TMimePart;
    PartCharset: TMimeChar; Raw: Boolean;
    PartEncoding: TMimeEncoding): TMimepart;
function AddPartTextFromFile(const FileName: String;
    const PartParent: TMimePart): TMimepart;
```



Binaire data kun je als bijlage toevoegen via:

```
function AddPartBinary(const Stream: TStream;
    const FileName: string;
    const PartParent: TMimePart): TMimepart;
function AddPartBinaryFromFile(const FileName: string;
    const PartParent: TMimePart): TMimepart;
```

Alle functies hebben als laatste argument PartParent. Dat is het multipart-deel waaraan de toe te voegen bijlage op zijn beurt toegevoegd moet worden. Alleen multipart-delen kunnen andere delen bevatten. Als je probeert een binary-part aan een ander binary-part te koppelen, zal dat een foutmelding opleveren.

Er zijn nog wat hulpfuncties voor het toevoegen van HTML en het toevoegen van complete e-mailberichten. Als alle onderdelen eenmaal aan een bericht zijn toegevoegd, moet het bericht in zijn totaliteit samengesteld worden (geencodeerd). Dat kan via de functie EncodeMessage.

Het beheer van de bijlagen gebeurt via een listbox (LBAttachments). Een pop-upmenu toont twee menu-items, een om een bijlage toe te voegen en een om de geselecteerde bijlage te verwijderen. Dezelfde acties kunnen via knopjes op de knoppenbalk worden uitgevoerd.

Het gebruik van de TMimeMess-klasse is eenvoudig te demonstreren. In de applicatie worden de bestandsnamen van de bijlagen in een lijst van strings bijgehouden: FAttachments.

De OnExecute-handler van de te verzenden actie wordt nu wat uitgebreider:

```
procedure TMainForm.AVerzendenExecute(Sender: TObject);
begin
    if not CheckSettings then
        Exit;
    if (FAttachments.Count>0) then
        DoSendMailAndAttachments(ETo.Text,
                                 ESubject.Text,MMail.,Lines)
    else
        DoSendMail(ETo.Text,ESubject.Text,MMail.Lines);
end;
```

Als er bijlagen zijn, wordt de functie DoSendMailAndAttachments opgeroepen, die dan het eigenlijke werk doet:

```
procedure TMainForm.DoSendMailAndAttachments(
    Const ATo,ASubject : String;
    Content : TStrings);
Var
    Mime : TMimeMess;
    P : TMimePart;
    I : Integer;
    B : Boolean;
    begin
Mime:=TMimeMess.Create;
    try
        // Zet enkele headers van het bericht.
        Mime.Header.ToList.Text:=ATo;
        Mime.Header.Subject:=ASubject;
        Mime.Header.From:=FSender;
        // Maak een multipart deel aan.
        P:=Mime.AddPartMultipart('mixed',Nil);
        // Als eerste deel de tekst van de mail toevoegen.
        Mime.AddPartText(Content,P);
        // Alle bijlagen toevoegen.
        For I:=0 to FAttachments.Count-1 do
```

Door op het pictogram voor het toevoegen van een bijlage te klikken, kun je een bestand selecteren dat je wilt meesturen.

```
            Mime.AddPartBinaryFromFile(FAttachments[I],P);
```

Na deze code is het MIME-bericht klaar. Het bericht dat verzonden moet worden, kan nu samengesteld worden door EncodeMessage op te roepen en met SendToRaw te versturen:

```
        // Bericht samenstellen
        Mime.EncodeMessage;
        // En verzenden met SendToRaw
        B:=SendToRaw(FSender,ATo,FSMTPHost,Mime.Lines,
                                    FSMTPUser,FSMTPPasswd);
        if not B then
            ShowMessage('Kon het bericht niet verzenden!')
        else
            ShowMessage('Bericht verzonden.');
    finally
        Mime.Free;
    end;
end;
```

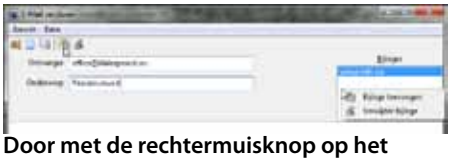
Nadat EncodeMessage is opgeroepen, bevat de property Lines van de TMimeMess-klasse het geëncodeerde bericht, inclusief de mailheaders. Dit bericht moet met SendToRaw verstuurd worden. Deze functie is als volgt gedefinieerd:

```
function SendToRaw(const MailFrom, MailTo,
                                        SMTPHost: string;
    const MailData: TStrings;
    const Username, Password: string): Boolean;
```

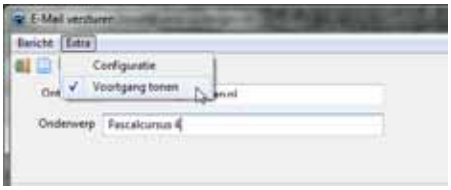
Deze functie verschilt van SendTo en SendToEx doordat er geen e-mailheaders worden samengesteld. De e-mailheaders moeten al in MailData aanwezig zijn. TMimeMess stelt deze headers dan samen in EncodeMessage.

Aanpassen

In de online wereld van vandaag behoren taken als het versturen van e-mails tot de standaard functionaliteit van veel programma's. Lazarus biedt in combinatie met Synapse een eenvoudige manier om e-mails te versturen, zoals we in dit artikel hebben duidelijk gemaakt. In dit geval hebben we dat met een duidelijke interface voor de in- en uitvoer gedaan, maar je kunt het afhandelen van e-mails ook geheel binnen je applicatie houden en het samenstellen, versturen en afvangen van fouten ook geheel



Door met de rechtermuisknop op het Bijlages-blok te klikken kun je ook rechtstreeks een bijlage toevoegen of weer verwijderen.



Als je het tonen van de voortgang aanvinkt…

intern houden zonder dat de gebruiker daar wat van merkt. Het basisprincipe blijft immers hetzelfde.

Bedenk ook dat we hier alleen de e-mail via SMTP naar de ingestelde mailserver versturen. Die mailserver is verder verantwoordelijk voor het daadwerkelijk versturen naar alle geadresseerden. Als daar iets fout loopt, bijvoorbeeld omdat een e-mailadres niet klopt, dan krijg je daar geen melding van.

… wordt tijdens het versturen van de mail getoond hoe ver dat gevorderd is.

We hebben als extraatje nog een voortgangsbalk ingebouwd. Dat is met name handig als je een e-mail met een grote bijlage wilt versturen om te zien dat er inderdaad wat gebeurt. Als je in de e-mailclient 'Extra / Voortgang tonen' aanvinkt, krijg je tijdens het versturen onderin de mail te zien hoe ver het versturen van je mail gevorderd is.

Als je dieper in de SMTP-materie wilt duiken, is het aan te raden in de broncodes in de map \synapse40\source\lib te kijken. Met name in smtpsend.pas, waar de klasse TSMTPSend beschreven wordt, zul je veel van de hier gebruikte code weer tegenkomen.

De volgende keer gaan we ons bezighouden met het omgaan met multimedia binnen Pascal, toegespitst op het gebruik van een webcam met alle mogelijkheden van dien. (nkr)

Cursus Pascal

Het doel van deze minicursus is dat je de basisbeginselen leert van het programmeren in Pascal met de ontwikkelomgeving Lazarus en dat je een aantal eenvoudige programma's hebt kunnen maken met behulp van componenten.

Deze cursus is geschreven in samenwerking met de Stichting Ondersteuning Programmeertaal Pascal.

Deel 1. De basis
Deel 2. Debuggen
Deel 3. Classes
Deel 4. E-mailen
– de netwerkmogelijkheden van Pascal aan de hand van SMTP
Deel 5. Multimedia
Deel 6. Databases

Literatuur

[1] Detlef Overbeek, Noud van Kruysbergen, "Hallo Wereld!", Programmeercursus Pascal, deel 1: de basis, c't 4/2012, p.102
[2] Siegfried Zuhr, Noud van Kruysbergen, Zoek de vout, Programmeercursus Pascal, deel 2: het debuggen, c't 5/2012, p.138
[3] Anton Vogelaar, Noud van Kruysbergen, Met klasse, Programmeercursus Pascal, deel 3: classes, c't 6/2012, p.138

Softlink 1207138