

# Envio automatizado de emails com Delphi e Outlook - Parte 2



**C**ontinuando a série de artigos sobre automação Outlook via Delphi, dessa vez demonstrarei alguns recursos mais avançados, como anexar arquivos e formatar mensagens em HTML.

Na edição 82, vimos como percorrer a tabela de funcionários do *Northwind* e, para cada funcionário cadastrado, gerar um e-mail com dados de uma mala direta, substituindo as variáveis pelos dados da tabela. Agora aperfeiçoaremos o exemplo para gerar e-mails em formato HTML. A idéia continua a mesma e as modificações serão mínimas.

---

**Atenção:** Será necessário que o leitor possua algum conhecimento prévio com a sintaxe da linguagem HTML, pois não entraremos em detalhes sobre as tags e códigos HTML. É necessário também o uso da biblioteca **Redemption Objects**, disponível em: [www.dimastr.com/redemption](http://www.dimastr.com/redemption)

---

## Criando uma página de modelo para o email

Vamos criar uma página HTML que servirá como base para o envio da mala direta. Você poderá utilizar qualquer editor de texto, como o Bloco de Notas, mas recomendo utilizar o Adobe Dreamweaver, por facilitar a criação. Crie o mesmo texto do artigo anterior e formate-o como desejar. Observe a **Figura 1**.

Mantemos as variáveis entre chaves, exatamente como no artigo anterior. Elas serão substituídas pelos dados da tabela de Funcionários do *Northwind*. Na **Listagem 1**, temos um código exemplo do modelo em formato HTML.

---

**Atenção:** Na mensagem modelo, uma imagem foi incluída na tag ``. Ao enviar uma mala direta, você deverá trocar o endereço da imagem para o endereço em que ela se encontra hospedada, por

exemplo: ``. Assim, quando o destinatário receber o e-mail, o Outlook realizará o download da imagem diretamente do servidor indicado.

## Alterando o projeto para enviar emails em HTML

No Delphi, abra o projeto do artigo anterior e configure a tela como a **Figura 2**.

Observe que retiramos o *Memo* que armazenava a mensagem em formato texto, e adicionamos um *Edit* (“edtArquivo”) que armazenará o caminho completo do modelo. O usuário poderá selecionar o modelo que quiser através do *Selecionar Arquivo* (“btnAbrir”).

Adicionaremos uma nova função na seção *private*, chamada *AbrirArquivo*, que será utilizada para abrir o arquivo de modelo e armazenar seu conteúdo em um *StringList*, para posterior substituição das variáveis e envio do e-mail.

Também criaremos uma nova variável, chamada *DirAplicativo*, para armazenar o diretório atual do aplicativo. Veja o código completo na **Listagem 2** (não esqueça de excluir o código onde consta o *Memo* removido anteriormente).

### Listagem 1. Código HTML da página de modelo

```
<html>
<style>
body{font-size:11px;font-family:Tahoma;margin:10px;}
#mensagem{border:1px solid #999999;}
#titulo{background:#F5D47C;padding:5px;
font-weight:bold;border:1px solid white;}
#imagem{float:left;margin-top:10px;
margin-right:10px;margin-left:5px;}
#corpo{padding:10px;}
#agradecimento{padding-left:10px;}
#rodape{padding-left:10px;padding-bottom:8px;}
#linha{padding:0px}
.titulo1{font-size:12px;font-weight:bold;}
.titulo2{font-size:11px;font-style:italic}
</style>

<body>
<div id="mensagem">
<div id="titulo">Prezado [TRATAMENTO] [NOME] [SOBRENOME],
</div>
<div id="imagem"></div>
<div id="corpo">
<br>estamos lhe enviando esta mensagem com o intuito
de divulgar nosso mais novo software para envio
automático de emails e mala direta.
<br><br><br>
<p>Podemos destacar diversas funções, entre elas:</p>
<li>Personalização de modelos;</li>
<li>Envio em formato HTML;</li>
<p>entre outros.</p>
<p>Caso queira saber mais, envie-nos um email, que entraremos em contato assim que possível.</p>
</div>
<div id="agradecimento">
Agradecemos sua atenção.
</div>
<div id="linha"><hr size="1" color="#999999"></div>
<div id="rodape">
<span class="titulo1">Sistemas</span>
<br>
</div>
</body>
</html>
```

Apenas uma parte pequena do código foi alterada. Trocamos a propriedade *Body* para *HTMLBody* e a propriedade *BodyFormat* para *olFormatHTML* no código do *OnClick* do botão que envia o e-mail. Além disso, criamos uma função (*AbrirArquivo*) para ler o conteúdo do arquivo HTML e jogá-lo dentro de um *StringList*, para manipulação interna das variáveis e envio dos e-mails.

Também adicionamos um botão para permitir ao usuário escolher uma página HTML, e na variável *DirAplicativo* armazenamos o diretório atual do aplicativo. Execute a aplicação e faça o teste.

## Anexando arquivos

Alteraremos o mesmo projeto para anexar um arquivo relacionado ao produto da mala direta (folder). Esse recurso poderá ser utilizado para empresas que desejam automatizar o processo de marketing de seus produtos e serviços.

Adicione um *Edit* (“edtAnexo”), um *Button* (“btnAnexar”) e na propriedade *Caption*, digite “Inserir Anexo”. Veja na **Figura 3** como ficou a aplicação.

Implemente o evento *OnClick* do botão de anexar, através da **Listagem 3**, que será responsável por criar uma janela para permitir ao usuário a escolha do anexo.

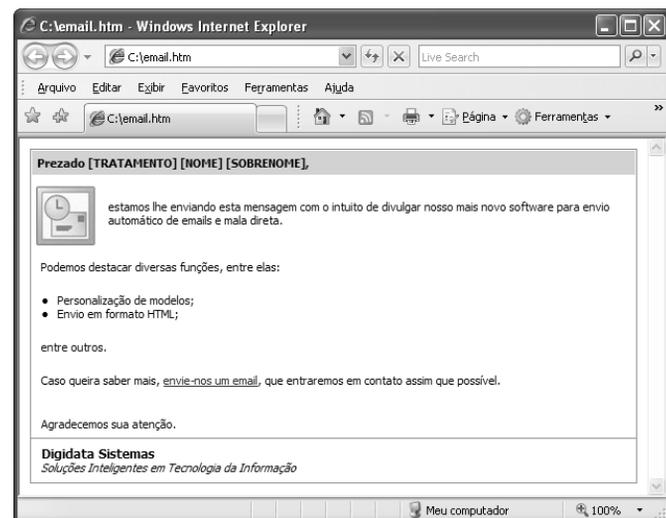


Figura 1. Visualizando o modelo da mensagem no navegador

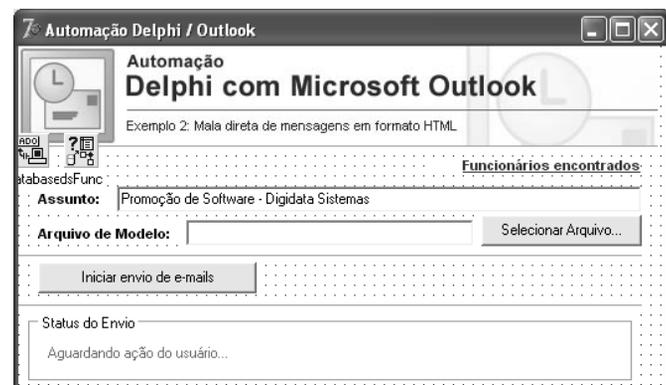


Figura 2. Tela principal do aplicativo modificado para utilizar modelos HTML

No evento *OnClick* do *Enviar*, após a linha *MI.HTMLBody := CorpoEmail*, adicione o código a seguir, que será responsável por incluir o anexo na mensagem.

```
if FileExists(edtAnexo.Text) then
  oItem.Attachments.Add(edtAnexo.Text);
```

Agora rode o aplicativo, adicione um arquivo anexo e clique no botão de enviar os e-mails. Observe que para cada e-mail gerado, o Outlook anexou o arquivo escolhido no aplicativo (**Figura 4**).

#### Listagem 2. Código modificado do aplicativo

```
...
private
  DirAplicacao: string;
  procedure AbrirArquivo(Arquivo: string;
    Lista: TStringList);
...
public
end;

var
  frmPrincipal: TfrmPrincipal;

Implementation

uses OleServer, OutlookXP, Redemption_TLB, ComObj;
{$R *.dfm}

procedure TfrmPrincipal.AbrirArquivo(Arquivo: string; Lista: TStringList);
var
  ArquivoTexto: TextFile;
  Conteudo: string;
begin
  Lista.Clear;
  AssignFile(ArquivoTexto, Arquivo);
  FileMode := fmOpenRead;
  Reset(ArquivoTexto);
  while not Eof(ArquivoTexto) do
  begin
    ReadLn(ArquivoTexto, Conteudo);
    Lista.Add(Conteudo);
  end;
  CloseFile(ArquivoTexto);
end;

...
procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
  dsFunc.Open;
  { Código novo }
  DirAplicacao := ExtractFileDir(Application.ExeName);
  edtArquivo.Text := DirAplicacao + '\email.htm';
end;

...
procedure TfrmPrincipal.btnAbrirClick(Sender: TObject);
var
  dlgAbrir: TOpenDialog;
begin
  dlgAbrir := TOpenDialog.Create(self);
  try
    dlgAbrir.InitialDir := DirAplicacao;
    dlgAbrir.Options := [ofFileMustExist];
    dlgAbrir.Filter :=
      'Páginas da Web (htm,html)|*.htm;*.html';
    dlgAbrir.FilterIndex := 2;
    if dlgAbrir.Execute then
      edtArquivo.Text := dlgAbrir.FileName;
  finally
    dlgAbrir.Free;
  end;
end;

{ Código modificado }
procedure TfrmPrincipal.btnEnviarClick(Sender: TObject);
var
  Outlook: TOutlookApplication;
  NmSpace: NameSpace;
  MI, oItem, MAPIFolder, sFolder: OleVariant;
  TextoEmail: TStringList;
  sAssunto, sCampo, sDados: string;
  CorpoEmail: WideString;
  a, b, c: integer;
begin
  lblInfo.Caption := 'Conectando ao Outlook...';
  Application.ProcessMessages;
  try
    Outlook := TOutlookApplication.Create(nil);
    Outlook.ConnectKind := ckRunningOrNew;
```

## Conclusão

Aperfeiçoamos o mesmo projeto com poucas alterações no código e adicionamos recursos para envio de mensagens em formato HTML e com anexos. No próximo artigo construiremos um banco de dados e um aplicativo para filtrar mensagens recebidas no Outlook e, através de filtros personalizados, armazenaremos as mensagens recebidas. Bons estudos e até a próxima. ■

```
Outlook.Connect;
except
  lblInfo.Caption :=
    'Erro ao conectar ao Outlook...';
  Application.ProcessMessages;
  Abort;
end;

try
  lblInfo.Caption := 'Buscando perfil de acesso...';
  Application.ProcessMessages;
  NmSpace := Outlook.GetNamespace('MAPI');
  NmSpace.Logon('', '', False, False);
  TextoEmail := TStringList.Create;
  sAssunto := edtAssunto.Text;
  { Para cada registro, enviar um e-mail }
  lblInfo.Caption := 'Percorrendo registro de funcionários...';
  Application.ProcessMessages;
  while not dsFunc.EOF do
  begin
    lblInfo.Caption := 'Gerando mensagem para: ' +
      dsFunc.FieldName('NOME').AsString;
    Application.ProcessMessages;
    { Limpando as variáveis }
    CorpoEmail := '';
    TextoEmail.Clear;
    AbrirArquivo(edtArquivo.Text, TextoEmail);
    { TextoEmail.Assign(memTexto.Lines); }
    { Localizando e substituindo variáveis entre chaves [] por dados }
    for a:= 0 to TextoEmail.Count-1 do
      for b:= 0 to dsFunc.Fields.Count-1 do
        begin
          sCampo := Uppercase(dsFunc.Fields[b].FieldName);
          sDados := dsFunc.FieldName(sCampo).AsString;
          if pos('[+' + sCampo + ']', TextoEmail[a]) > 0
            then
              TextoEmail[a] := SubstituirCampoTexto(
                TextoEmail[a], '[' + sCampo + ']', sDados);
        end;
        oItem := Outlook.CreateItem(0);
        MI := CreateOleObject('Redemption.SafeMailItem');
        MI.Item := oItem;
        { O formato agora é HTML }
        MI.Item.BodyFormat := olFormatHTML;
        MI.Recipients.Add(dsFunc.FieldName('EMAIL').AsString);
        MI.Subject := sAssunto;
        for c:= 0 to TextoEmail.Count-1 do
          CorpoEmail := CorpoEmail + #13 + TextoEmail.Strings[c];
          { Alteramos aqui para HTMLBody }
          MI.HTMLBody := CorpoEmail;
        MI.Send;
        dsFunc.Next;
      end;
    lblInfo.Caption :=
      'Concluindo criação de mensagens...';
    Application.ProcessMessages;
    MAPIFolder := Outlook.Session.GetDefaultFolder(olFolderInbox);
    sFolder := CreateOleObject(
      'Redemption.MAPIFolder');
    sFolder.Item := MAPIFolder;
    sFolder.Display;
  finally
    lblInfo.Caption := 'Limpendo variáveis...';
    Application.ProcessMessages;
    try
      NmSpace.LogOff;
    except
    end;

    try Outlook.Disconnect;
    except
    end;
  end;

  lblInfo.Caption := 'Operação concluída com sucesso...';
  Application.ProcessMessages;
end;
end.
```

Listagem 3. Evento OnClick do Inserir Anexo

```

var
  dlgAnexo: TOpenDialog;
begin
  dlgAnexo := TOpenDialog.Create(self);
  try
    dlgAnexo.InitialDir := DirAplicacao;
    dlgAnexo.Options := [ofFileMustExist];
    dlgAnexo.Filter :=
      'Arquivos de Folder (pdf,doc,xls)|*.pdf; '+
      '*.doc;*.xls';
    dlgAnexo.FilterIndex := 2;
    if dlgAnexo.Execute then
      edtAnexo.Text := dlgAnexo.FileName;
  finally
    dlgAnexo.Free;
  end;
end;

```

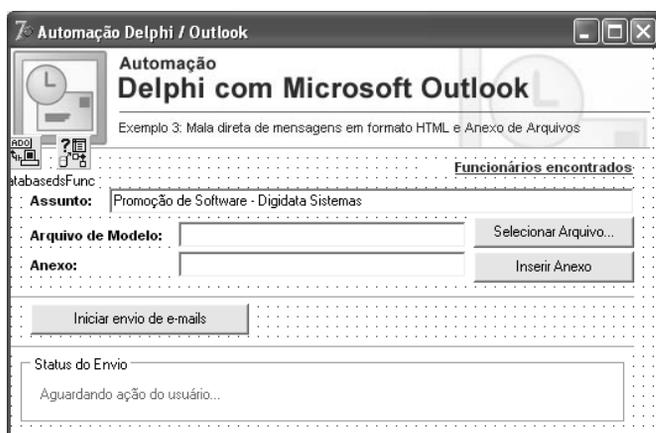


Figura 3. Interface principal após as alterações efetuadas



## Links

## Dicas de automação em Delphi

[delphi.about.com/od/kbcontrolole/OLE\\_Automation\\_using\\_Delphi.htm](http://delphi.about.com/od/kbcontrolole/OLE_Automation_using_Delphi.htm)  
[djgate.freeseerve.co.uk/Automation.htm](http://djgate.freeseerve.co.uk/Automation.htm)

## Dicas de automação Outlook via Delphi

[www.djgate.freeseerve.co.uk/AutoOutl.htm](http://www.djgate.freeseerve.co.uk/AutoOutl.htm)

## Dúvidas

E-mail: [mirilo.filho@asa-sistemas.com](mailto:mirilo.filho@asa-sistemas.com)

Web site: [www.asa-sistemas.com](http://www.asa-sistemas.com)

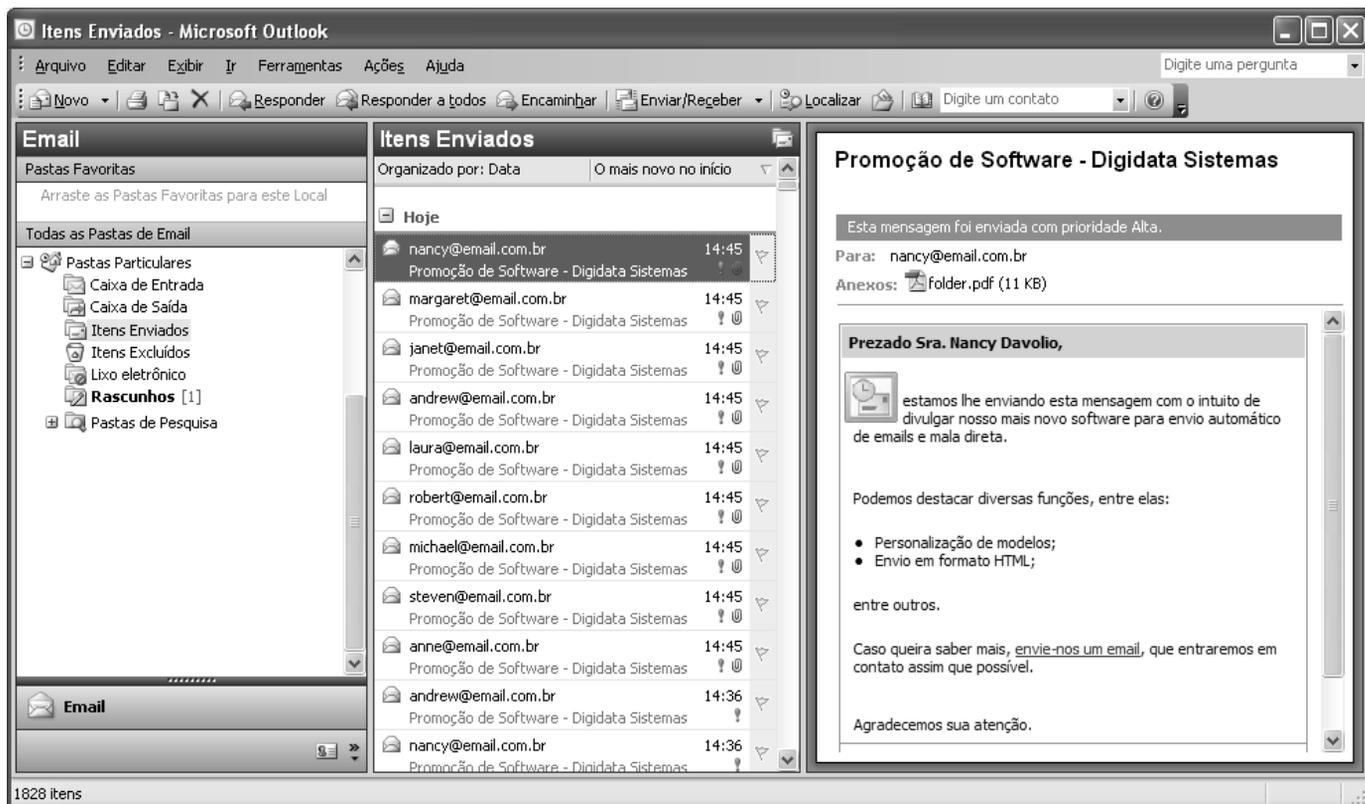


Figura 4. Mensagens enviadas no Outlook (observe o anexo)