



Acessando Outlook através do Delphi – itens de e-mail.

Plataformas: **Delphi 7, Outlook 2003.**

Apesar de o aplicativo ter sido desenvolvido utilizando-se o Outlook 2003, deverá funcionar com qualquer versão do Outlook a partir do Outlook 97, pois o modelo de Objetos manteve a sua interface de acesso através das versões.

Este não é um tutorial elaborado, passo a passo, do tipo que cobre todas as facetas da integração com o Outlook, até porque o tempo anda meio escasso neste início de ano (ainda bem, sinal de que temos trabalho para poder pagar as contas né?), o que pretendo é mostrar o caminho para que vocês possam explorar a biblioteca de objetos e coleções do Outlook e interagir com ela, para poder trabalhar com estas informações dentro do seu aplicativo.

Este documento vai se resumir aos itens de e-mail, em outro documento abordarei os demais itens do outlook, como contatos, calendário, tarefas, anotações, etc.

Bem, garotas e garotos, chega de papo furado e vamos à luta, porque a fila anda e a roleta gira.

Primeiro de tudo, você precisará do Outlook instalado na sua máquina, qualquer versão serve, é apenas para você efetuar o desenvolvimento ir testando à medida que vai desenvolvendo.

Vamos ao Delphi, iniciando um novo projeto.

- Inicie um novo projeto no Delphi.
- Para facilitar a programação, sugiro que você importe a Type Library do Outlook para um arquivo que usará no seu aplicativo, dessa maneira poderá usar as enumerações do Outlook para manipular os objetos.
- Você poderá usar os objetos, métodos e propriedades da Type Library, neste tutorial optei por usar automação ActiveX direto, usando o método “*mão na graxa*”, sabe como é, primeiro a gente faz da maneira mais dura, depois vai descobrindo o caminho fácil. Penso que essa é a melhor maneira de vocês irem pegando intimidade com a coisa.
- No Delphi, acesse o menu **Project -> Import Type Library**;
- Escolha o item **Microsoft Outlook xx Object Type Library**, onde xx representa a versão que estará instalada na sua máquina, no meu caso é 11.0
- Selecione este item, e clique em **Create Unit**, será gerada uma unit chamada Outlook_TLB.pas, que você irá usar no seu aplicativo.
- Vamos usar este item, na cláusula uses, da seção implementation, vamos incluir esta unit Outlook_TLB.
- Agora já temos toda a coleção de constantes, e itens do Outlook.
- Inclua também na seção uses principal, da parte interface, duas units: **Variants** e **ComObj**;
- Vamos declarar as variáveis que vamos usar para acessar o Outlook.
- Primeiro de tudo, você precisa do aplicativo Outlook para poder acessar, então vamos declarar uma variável chamada OApI (**Outlook Application**), ela irá se referenciar ao próprio aplicativo Outlook;

O modelo de objeto do Outlook fornece todas as funcionalidades necessárias para a manipulação de dados armazenados em pastas do Outlook e a capacidade de controlar vários aspectos da interface de usuário do Outlook.

Para iniciar uma sessão de automação do Outlook, você pode usar a vinculação precoce ou tardia. A vinculação tardia usa a função **CreateOleObject** para inicializar o Outlook, e será a que usaremos neste documento.



Por exemplo, o código a seguir define uma variável de objeto para o objeto Application do Outlook, que é o objeto que está no nível mais alto do modelo de objeto do Outlook. Todos os códigos de automação devem primeiro definir um objeto **Application** do Outlook para poder acessar qualquer outro objeto do dito cujo carará sanguinolento.

```
var
  OApl : Variant;
begin
  OApl := CreateOleObject('Outlook.Application');
end;
```

Nosso aplicativo irá interagir com os dados armazenados no Outlook, por isso vamos ter que usar o objeto **MAPI**.

O Outlook armazena todas as suas informações em pastas **MAPI** (Message application programmer interface – Interface de programação de aplicativos de mensagens), este é o segundo nível mais alto da hierarquia de objetos do Outlook, o nível mais alto é, como já foi dito, o próprio objeto Application.

A **MAPI** é acessada através de um objeto NameSpace, este objeto é que faz referência à esta interface, veja exemplo abaixo:

```
var
  OApl, ONS: Variant;
begin
  OApl := CreateOleObject('Outlook.Application');
  ONS := OApl.GetNameSpace('MAPI');
end;
```

A partir deste ponto, quem irá nos fornecer os objetos que desejamos, no nosso caso será a caixa de entrada, é o objeto NameSpace, que agora tem a referência à coleção **MAPI**.

Agora podemos definir os objetos que iremos usar para conseguir acessar a caixa de entrada. Criei as variáveis com nomes bem fáceis de entender, todas começam com **O**, de Outlook (viram como sou inteligente?), assim temos:

OPasta – Que acessa a pasta que desejamos no Outlook, no nosso caso será a caixa de entrada.

OMensagens – Este objeto irá referenciar todas as mensagens e eventuais subpastas da caixa de entrada. Sim, isso mesmo, se você tiver subpastas, este objeto fará referencia a elas também, além das mensagens propriamente ditas.

OMensagensFiltradas – Aqui iremos referenciar apenas as mensagens que forem recuperadas pelo filtro que vamos aplicar.

```
var
  OApl, ONS, OPasta, OMensagens, OMensagensFiltradas: Variant;
begin
  OApl := CreateOleObject('Outlook.Application');
  ONS := OApl.GetNameSpace('MAPI');
  OPasta := ONS.GetDefaultFolder(OIFolderInbox);
  OMensagens := OPasta.Items;
end;
```

Com isto já temos todas as mensagens da caixa de entrada (OIFolderInbox), resta apenas conseguir filtrar as ditas cujas.

Fazer isto é fácil, os objetos deste nível possuem um método chamado **Restrict**, que pode ser utilizado para fazer uma filtragem nos itens, você deverá montar um filtro, que é uma string, esta string é o argumento da função **Restrict**, e vai limitar o conjunto de resultados.

Atenção para o seguinte:



Se você estiver usando campos definidos pelo usuário como parte de uma cláusula **Restrict**, esses campos deverão existir na pasta. Caso contrário, o código gerará um erro indicando que o campo é desconhecido, ou seja... vai dar cacaca, e do tipo fedorento.

O método **Restrict** pode ser aplicado a quase todos os campos de informação dos itens, tais como nome do contato, data do email, remetente, assunto, etc. Porém você não pode usar este método com as seguintes propriedades, ou vai ter um erro. Cacaca fedorenta de novo, evite o cheiro ruim.

Body	LastFirstNoSpaceCompany
Categories	LastFirstSpaceOnly
Children	LastFirstSpaceOnlyCompany
Class	LastFirstNoSpaceAndSuffix
Companies	MemberCount
CompanyLastFirstNoSpace	NetMeetingAlias
CompanyLastFirstSpaceOnly	NetMeetingAutoStart
ContactNames	NetMeetingOrganizerAlias
Contacts	NetMeetingServer
ConversationIndex	NetMeetingType
DLName	RecurrenceState
Email1EntryID	ReplyRecipients
Email2EntryID	ReceivedByEntryID
Email3EntryID	ReceivedOnBehalfOfEntryID
EntryID	ResponseState
HTMLBody	Saved
IsOnlineMeeting	Sent
LastFirstAndSuffix	Submitted
LastFirstNoSpace	VotingOptions
AutoResolvedWinner	DownloadState
BodyFormat	IsConflict
InternetCodePage	MeetingWorkspaceURL
Permission	

Por exemplo, para filtrar apenas os e-mails com data de envio posterior a 01/01/2007, usamos o método **Restrict**, da seguinte maneira:

```
OMensagensFiltradas := Omensagens.Restrict("[ReceivedTime] >= ' + QuotedStr('01/01/2007'));
```

O método **restrict** será aplicado à pasta de mensagens, e teremos uma nova referência somente as mensagens filtradas (viram a importância de nomes sugestivos para as variáveis? Ainda mais para eu que sou meio esquecido).

Algumas observações sobre os argumentos do método **Restrict**.

Datas:

Embora as datas e horas sejam normalmente armazenadas com um formato de data, o método **Restrict** exige que a data e a hora sejam convertidas para uma representação de seqüência de caracteres, o formato de data neste caso será sempre o que estiver definido no painel de controle do Windows.

Este método suporta que você informe também a hora e minuto junto com a data, permitindo uma filtragem mais precisa ainda.

Operadores Boolean

Os operadores **Boolean**, TRUE/FALSE, YES/NO, ON/OFF etc. não devem ser convertidos em uma seqüência de caracteres, ou seja, não use a função **QuotedStr**, informe-os de forma textual, o modelo de objetos de Outlook se encarrega de interpretar corretamente. Por exemplo, para determinar se o diário está habilitado para contatos, você pode usar este filtro:

```
StrFiltro = '[Journal] = True';
```



Neste caso, a propriedade **Journal**, indica que o item diário está habilitado para guardar referência aos contatos.

Atenção: Se você usar aspas como delimitadores com campos **Boolean**, uma seqüência de caracteres vazia localizará itens cujos campos sejam **False** e todas as seqüências de caracteres não-vazias localizarão os itens cujos campos sejam **True**.

Keywords (ou Categorias)

O campo Categories é do tipo palavras-chave, que se destinam a conter diversos valores. Quando acessado através de programação, o campo Categories se comporta como um campo Texto e a seqüência de caracteres deve coincidir exatamente. Os valores na seqüência de caracteres são separados por uma vírgula e um espaço. Isso geralmente significa que usar o método Restrict em um campo de palavras-chave se ele contiver mais de um valor não irá produzir o resultado que você espera.

Por exemplo, se houver um contato na categoria Fornecedor e um contato nas categorias Fornecedor e Cliente, você não poderá usar facilmente os métodos Find e Restrict para recuperar todos os itens que estejam na categoria Fornecedor.

Em vez disso, percorra todos os contatos na pasta e use uma função (como Pos(), por exemplo) para testar se a seqüência de caracteres "Fornecedor" está contida no campo de palavras-chave inteiro.

Uma exceção possível seria se você limitasse o campo Categories a dois ou poucos valores. Então, você poderia usar os métodos Find e Restrict com o operador lógico OR para recuperar todos os contatos de Fornecedor. Por exemplo (em pseudocódigo): "Fornecedor" OR "Fornecedor, Clientel" OR "Cliente, Fornecedor". As seqüências de caracteres de Category não diferenciam maiúsculas de minúsculas.

Este negócio de não diferenciar maiúsculas de minúsculas me lembrou uma vez, eu fazendo um curso de C, lá nos tempos do bit lascado, e o professor (muito esperto) resolveu traduzir o termo "case sensitive" e disse que o C era **sensível ao caso**.

Integer

Você pode procurar por campos Integer com ou sem aspas como delimitadores. Os filtros a seguir localizarão contatos que foram criados usando o Outlook 2000:

```
StrFiltro = "[OutlookInternalVersion] = 92711";
```

```
StrFiltro = "[OutlookInternalVersion] = \"92711\"";
```

Operadores lógicos:

O método **Restrict** permite que você use os operadores lógicos **AND**, **OR** e **NOT** para combinar um filtro mais complexo.

Propriedades do objeto MailItem:

Actions

AlternateRecipientAllowed

Application

Attachments

AutoForwarded

AutoResolvedWinner

BCC

BillingInformation

Body

BodyFormat

Categories

CC

Class

Companies

Conflicts

ConversationIndex

ConversationTopic

CreationTime

DeferredDeliveryTime



DeleteAfterSubmit
DownloadState
EnableSharedAttachments
EntryID
ExpiryTime
FlagDueBy
FlagIcon
FlagRequest
FlagStatus
FormDescription
GetInspector
HasCoverSheet
HTMLBody
Importance
InternetCodepage
IsConflict
IsIPFax
ItemProperties
LastModificationTime
Links
MarkForDownload
MessageClass
Mileage
NoAging
OriginatorDeliveryReportRequested
OutlookInternalVersion
OutlookVersion
Parent
Permission
ReadReceiptRequested
ReceivedByEntryID
ReceivedByName
ReceivedOnBehalfOfEntryID
ReceivedOnBehalfOfName
ReceivedTime
RecipientReassignmentProhibited
Recipients
ReminderOverrideDefault
ReminderPlaySound
ReminderSet
ReminderSoundFile
ReminderTime
RemoteStatus
ReplyRecipientNames
ReplyRecipients
Saved
SaveSentMessageFolder
SenderEmailAddress
SenderEmailType
SenderName
Sensitivity
Sent
SentOn
SentOnBehalfOfName
Session
Size
Subject
Submitted
To
UnRead



UserProperties VotingOptions VotingResponse

As propriedades em si, são praticamente autoexplicativas, e não é difícil usá-las para se montar os filtros que se quer, ou para recuperar informações dos itens.

Estes são os métodos do objeto MailItem:

ClearConversationIndex
Close
Copy
Delete
Display
Forward
Move
PrintOut
Reply
ReplyAll
Save
SaveAs
Send
ShowCategoriesDialog

Um bom ponto de apoio, para você verificar sobre o modelo de objetos do Outlook e como programar usando este framework, é o próprio Outlook, inicie o editor de Visual Basic, ou o editor de macros, e veja a referência de objetos do outlook, você terá todo o modelo de objetos para navegar, descobrir seus métodos, propriedades e eventos.

Note que, se a pasta do outlook (o arquivo *.pst), estiver protegido por senha, o aplicativo irá solicitar a senha automaticamente para o usuário. Nas versões mais novas do Outlook, será necessário configurar as diretivas de segurança para poder acessar a caixa de entrada sem problemas.

Bom caríssimos, por enquanto é só. O próximo artigo será como fazer um plugin e instalar o seu aplicativo no próprio Outlook, de maneira que não dependa de rodar o programa externamente.

Os fontes do aplicativo de exemplo, podem ser baixados no site da Spectrus, em www.spectrus.com.br, na parte de artigos e dicas.

Vou ficando por aqui, já é tarde e amanhã tenho que viajar para atender um cliente, e só consegui vôo saindo de Congonhas as 07:30, e eu simplesmente **detesto acordar cedo**.

Espero que este artigo e o aplicativo que o acompanha sirva para você começar a mexer no Outlook.

Até a próxima, ou Intel+.

Amilton Maciel (o AcidBytes).

