## **Resumo**

no texto sobre POO POC PE fala no fim sobre componentização que é, através do acoplamento de “programas como componentes” em outros programas existentes. ou seja, colocar no programa C o componente A ou B, sem que o programa C saiba como A ou B foi feito préviamente, tomando "consciência" de seu código apenas quando está lendo

## 

## **Introdução ao COM (Component Object Model)**

O COM (Component Object Model) é um padrão de interface binária introduzido pela Microsoft (r) em 1993 a fim de permitir o acoplamento entre aplicações independente da linguagem em que estas aplicações têm sido desenvolvidas

O termo COM incorpora, em muitas ocasiões, diversos outros termos tais como ActiveX, OLE, OLE Automation, COM+.

É comum encontrá-los em textos técnicos como "sinônimos" refernido-se ao mesmo conceito de interface binária.

## **Essência do COM**

Em essência, o COM é uma forma neutra, independente de linguagem, de se implementar objetos que podem ser utilizados em diferentes ambientes e através de fronteiras entre máquinas (servidores).

A aplicação mais popular de objetos COM é o de objetos de negócios em aplicações distribuídas.

## **Reutilização de objetos**

Componentes COM permitem o uso/reuso de objetos sem o conhecimento prévio de sua implementação (normalmente conhecida em tempo de execução apenas).

Isso é possível com o uso de técnicas de instanciação do tipo "Late Binding".A interface exposta pelo objeto em questão é conhecida em tempo de execução or meio da função QueryInterface()

Apesar das interfaces padrão terem suas implementações em diversas plataformas, o COM é utilizado principalmente com o MS Windows.

Com o advento do .NET Framework, a plataforma de aplicações sofreu uma mudança significativa mas, devido ao grande número de aplicações legadas que fazem uso do padrão COM, o .NET Framework prevê o suporte à este padrão tanto para consumo dos componentes previamente desenvolvidos quanto para a criação de novos componentes que precisam ser atualizados (regras de negócios) sem perda da compatiblidade com as aplicações legadas.

Outro fator relevante para a manutenção de compatibilidade com o padrão COM/COM+ é o suporte às aplicações altamente distribuídas e escaláveis.

## **O COM+**

Em aplicações de grande porte, componentes de negócio estão normalmente separadas da interface de usuário e são consumidos também por integração em outros sitemas. Em sistemas distribuídos, é comum encontrar componentes no padrão COM em um servidor de aplicações específico para instanciação e chamadas remotas.

O responsável por gerenciar este volume grande de chamads é o Component Services, que é uma evolução do MTS do Windows 2000. O padrão COM+ é, na verdade a aliança do Component Object Model com a aplicação de gerenciamento, o Component Services.

A decisão por utilizar o COM+ afeta significativamente o desing da aplicação.

Algumas perguntas devem ser respondidas em pleno detalhe antes da decisão da publicação de um componente COM no Componente Services, a saber:

· Qual a escala (tamanho) de usuários, dados e transações a minha aplicação deverá suportar?

· O servidor de componentes deverá ser isolado das demais camads da aplicação?

· Será necessário o controle de segurança em nível de perfil de usuário para execução das transações de negócio servidas pelo componente?

· É requisito utilizar balanceamento de carga para execução dessas transações de forma distribuída?

Do ponto de vista do desenvolvedor (programador) o escopo deste treinamento é como criar componentes COM que possam ser consumidos por aplicações legadas ou possam ser adicionadas ao Component Services utilizando o .NET Framework e a linguagem de programação C#.Topic:

## **Interfaces, Herança e IUnknown**

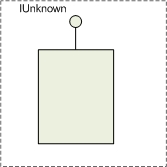
****

Figura 1 IUnknown

Para garantir tanto a interoperabilidade quanto a segurança dos componentes, o acesso às funcionalidades expostas é dado a partir da exposição de uma interface, um contrato de assinatura de métodos, propriedades e membros.

Nota: Interface é apenas uma determinação de “contrato” de implementação, isto é, as chamadas de um método em um objeto respeitarão o padrão de nome, parâmetros e tipo de dado retornado.

Herança de interface

Vale lembrar ou saber que HERANÇA no mundo COM não significa NECESSARIAMENTE, reutilização de código. São explicitadas, por padrão, apenas os “contratos” ou interfaces são visíveis aos consumidores do objeto COM. Em resumo, se uma interface herda de outra interface, isso indica que ela inclui os métodos (na verdade os contratos) definidos pela outra Interface.

Algumas interfaces são pré definidas no modelo COM. A mais importante delas é a interface Iunknown, que contém três métodos, a saber:

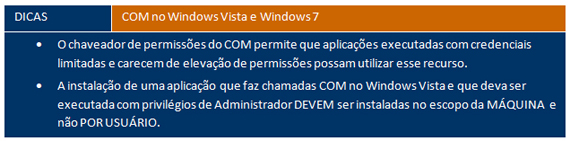
· QueryInterface

· AddRef

· Release

Todos os objetos COM devem implementar esta interface pois é o meio de prover meios de navegar entre as interfaces implementadas pelo objeto através do método QueryInterface e gerenciar o ciclo de vida do objeto por meio de AddRef e Release.[[1]](http://www.linhadecodigo.com.br/artigo/2864/o-component-object-model.aspx#_ftn1)

Em C# (.NET) você não vai ‘implementar’ essas funcionalidades explicitamente. Isso é feito automaticamente pelo compilador quando você aciona que um determinado assembly é “COM Visible”.



## **Como Criar classes COM usando o C#**

O tema implicito neste treinametno é interoperabilidade que, basicamente consiste em manter compatibilidade e comunicação com sistemas legados criados sobre plataformas anteriores ao .NET Framework. Em muitos casos, há inviabilidade de migrar todos os sistemas para uma nova plataforma pois os requisitos de negócio de uma empresa exigem que esta não pare suas operações por causa de um sistema. Há muitos sistemas críticos para realização de negócios nos mais diversos tipos de tecnologias. Uma delas é o modelo COM.

Este tutorial apresenta um pequeno caminho das pedras para criação de bibliotecas a partir de um assembly escrito em C# e que possa ser, por exemplo, consumido por uma aplicação criada no padrão Win32 (MFC ou ATL) ou ainda em aplicações criadas com o ASP 3.0 ou Visual Basic 6.0.

Crie um projeto do tipo ClassLibrary

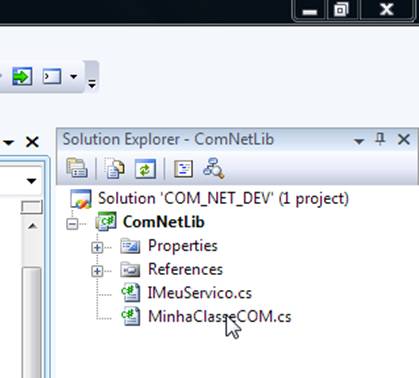


Figura 2 Projeto.

Procure, então, a partir do item Properties, o arquivo AssemblyInfo.cs e certifique-se de que o valor da propriedade ComVisible esteja marcado como True. Isso é fundamental para que as aplicações legadas consigam acessar os elementos do seu assembly.

[assembly: ComVisible(true)]

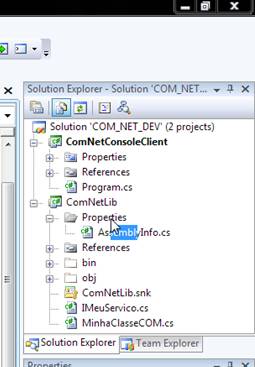


Figura 3 Localização do AssemblyInfo.cs

Lembre-se de que, para segurança de sua aplicação, os acessos às classes deve ser realizado por meio de interfaces. Neste tutorial, definimos uma interface denominada IMeuServico conforme o código abaixo:

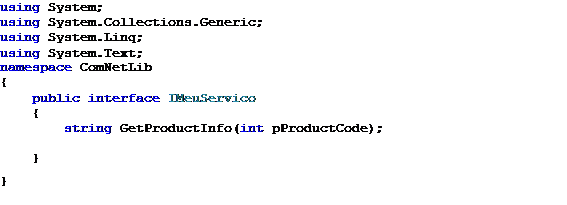


Figura 4 Interface

De forma opcional, mas recomendada, você pode atribuir um GUID para cada classe a ser exposta por COM. Para isso, utilize a ferramenta GUID Generator (Em TOOLS|CreateGUID), como ilustra a figura a seguir

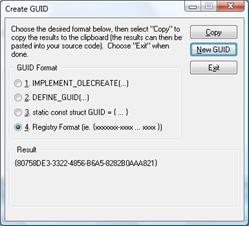


Figura 5 Gerador de GUID

Isso é feito através do atributo de classe Guid(),



Figura 6 Atributo de classe GUID

Sob o Namespace System.Runtime.InteropServices.

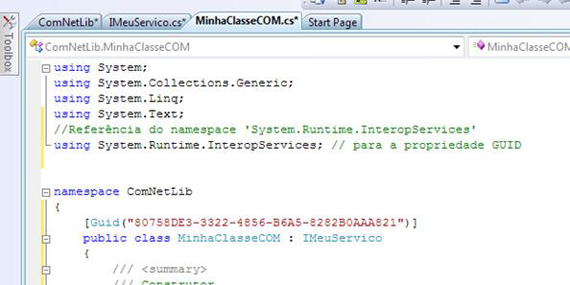


Figura 7 System.Runtime.InterpoServices

A partir disso, codifique sua classe com todas as funcionalidades previstas, em nosso exemplo:

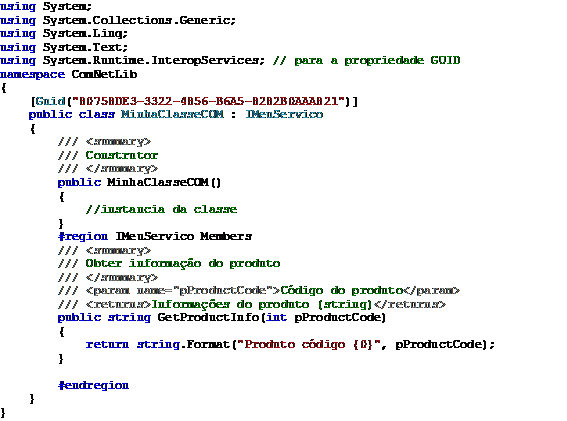


Figura 8 Classe COM Exemplo

Algumas configurações de projeto são necessárias à correta exposição de componentes no padrão COM quando se utiliza o .NET para implementação desses componentes, a saber:

· Marcar a configuração “Register for COM interop” na aba BUILD.

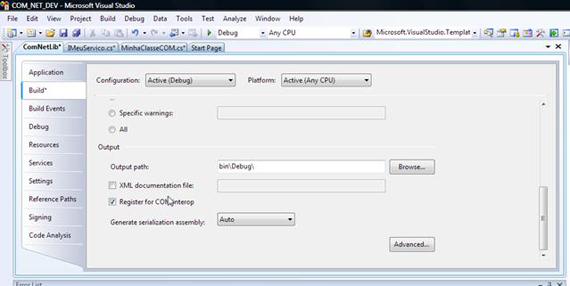


Figura 9 Propriedade do projeto Build

Para facilitar o processo de construção, agregar um pequeno script para automatizar a ‘criação’ e registro do assembly (biblioteca, no mundo COM, um assembly ainda se chama Biblioteca) após sua compilação. (evento Post-Buid)

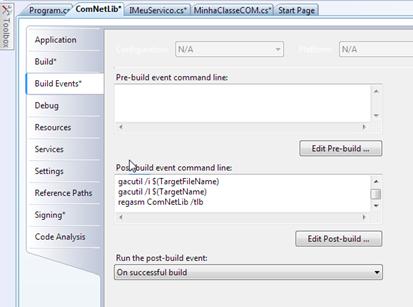


Figura 10 Build Events

Um exemplo de script é o que segue:

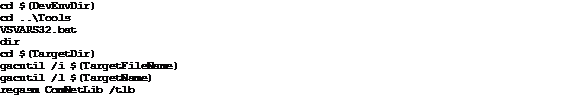


Figura 11 Modelo de script Post-Build

E ‘assinar’ o seu assembly, agregando uma “chave de nome forte” ao mesmo. Para que ele possa ser Registrado tanto no Global Assembly Cache (.NET) quanto no Registry por meio do regasm.exe

Isso é feito a partir da aba Signing do painel de configurações do projeto no Visual Studio.

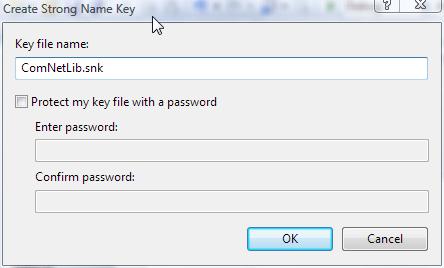


Figura 12 Criação do arquivo snk

Se estiver usando o Windows Vista, pode ocorrer erro de permissão para o registro. Neste caso, reinicie o Visual Studio como Administrador ou desabilite o UAC.

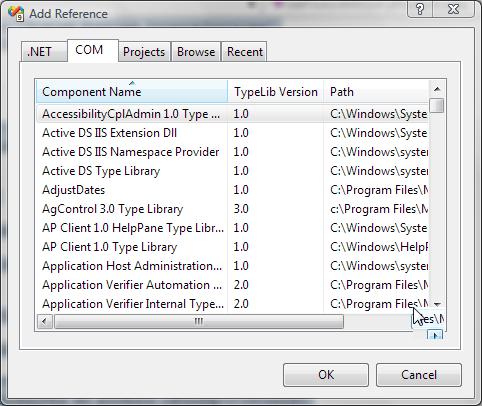


Figura 13 Erro de compilação no Windows Vista.

## **Como consumir uma “biblioteca” COM em .NET?**

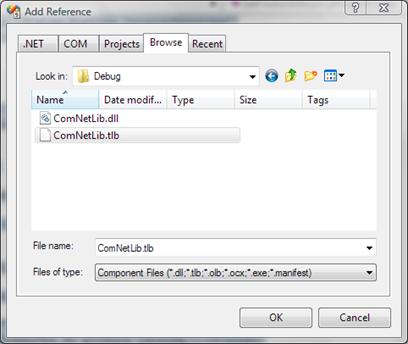
## **Early Binding**

O consumo por referência direta se dá da mesma forma como um assembly é referenciado num projeto criado no Visual Studio, por meio do menu Add Reference.



Os componentes listados nessa janela são os componentes (dlls Win32, OLE, MFC, ATL,etc) que estão devidamente cadastradas no Registry.

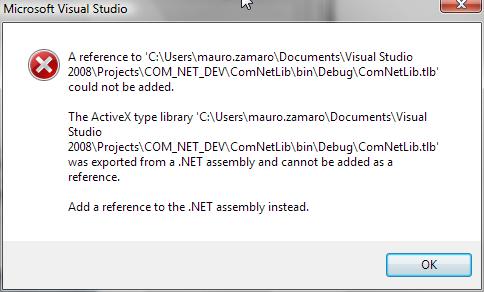
Quando um componente COM é adicionado a um projeto, o Visual Studio gera um proxy de interoperabilidade para ‘traduzir’ as interfaces expostas para consumo direto no código C# de forma transparente ao desenvolvedor.



Quando acionamos como referência um componente do tipo tlb (Type Libraries) e esta ter sido implementada pelo .NET Framework, o Visual Studio não precisa realizar a conversão de proxy, ele identifica que sua origem é um Assembly e solicita que seja usada a versão original Managed Code.

O arquivo tlb será utilizado apenas para referências em aplicações que foram construída em outras plataformas (VB6, ASP 3.0, Delphi, etc)

O aviso dado pelo Visual Studio é o seguinte:



## **Late Biding**

Uma outra maneira de consumir objetos COM é com o auxílo do Namespace System.Reflection. Neste caso, utilizamos as informações do registro desse componente no Registry para realizar a leitura, instanciação e chamadas (Invoke) dos métodos expostos pelas interfaces do componente.

Um exemplo de consumo por meio desta técnica é o que segue:

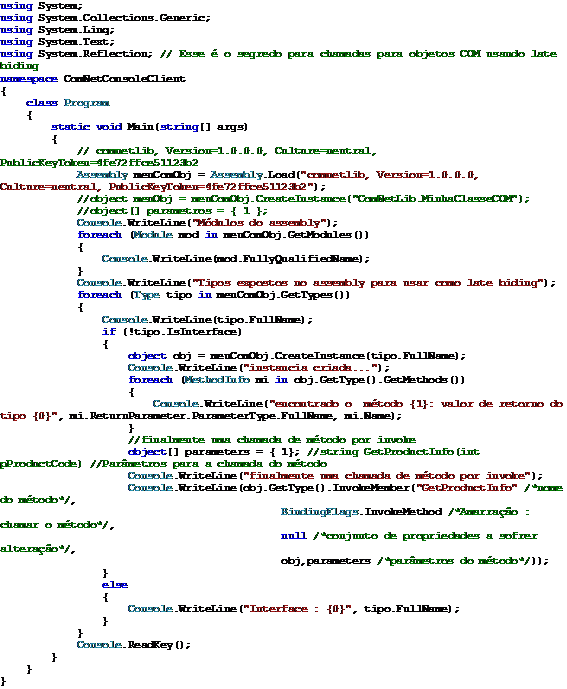
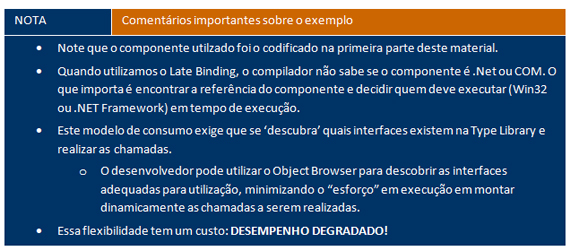


Figura 14 Exemplo de consumo com Late Binding



## **P/Invoke**

O Platform Invocation Services permite que uma aplicação .NET acesse código não gerenciado (padrão COM) e é utilizado principalmente para acesso à partes da API do Windows cuja funcionalidade não existe no .NET Framework

Para utilização do P/Invoke é necessário que você conheça MUITOo componente (ou a parte dele) que deseja utilizar. Isso pode ser resolvido com o Object Browser do Visual Studio como mostra a figura

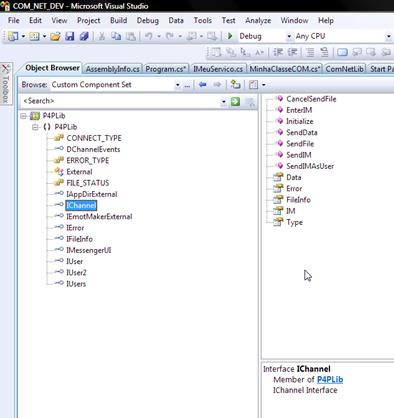


Figura 15 O Object Browser

Esta opção de interoperabilidade é indicada para funções da API que não estão disponíveis no .NET Framework. As principais APIs utilizadas são:

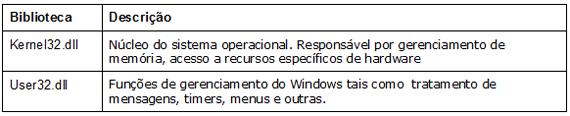


Figura 16 Bibliotecas da API Windos mais comuns para o P/Invoke

### **Como utilizar?**

Basta utilizar a propriedade de método DllImport, disponível no namespace System.Runtime.InteropServices conforme ilustra o exemplo abaixo:

Read more: <http://www.linhadecodigo.com.br/artigo/2864/o-component-object-model.aspx#ixzz6rWVpdmsZ>