

Criação de Objetos ASP em Delphi 5.0

Por Adenilton Rodrigues
Aden@aden.com.br
Belo Horizonte – MG
2001

Este documento pode ser livremente copiado e distribuído. Sua modificação só será permitida mediante contato prévio com o autor.

Delphi é marca registrada da Borland Inc. www.borland.com
Windows, PWS, IIS, ASP São marcas registradas e produtos da Microsoft Inc. www.microsoft.com

Como criar um objeto ASP em Delphi 5.0

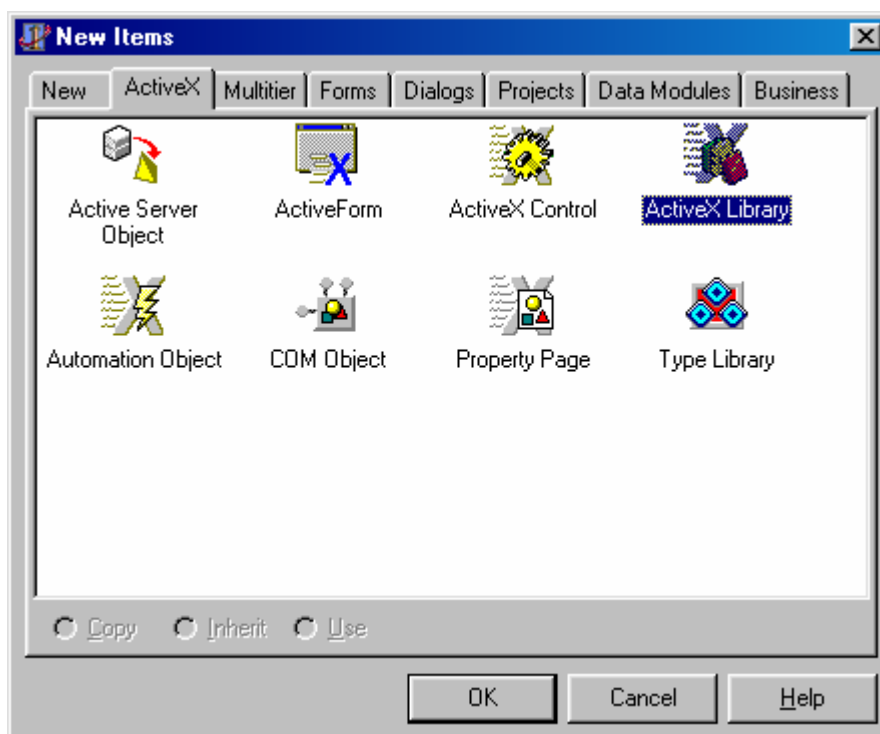
Criar objetos ASP em Delphi é uma tarefa relativamente simples. Esse tutorial demonstrará de forma rápida a técnica utilizada pelo Borland Delphi 5.0 para gerar arquivos DLL que poderão ser instanciados e manipulados dentro de scripts ASP.

Apresentarei um componente bastante simples cuja função será apresentar uma mensagem "Hello World" no browser.

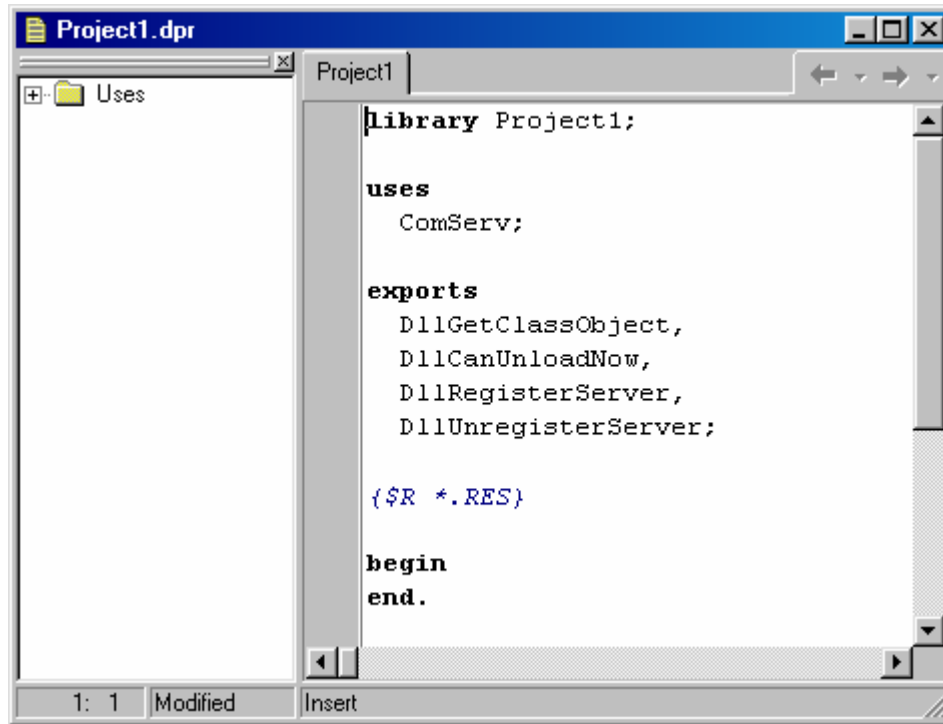
Vamos iniciar a criação da DLL.

Abra o Delphi. Se ele abrir com um projeto novo, feche-o usando a opção "Close All" do menu "File".

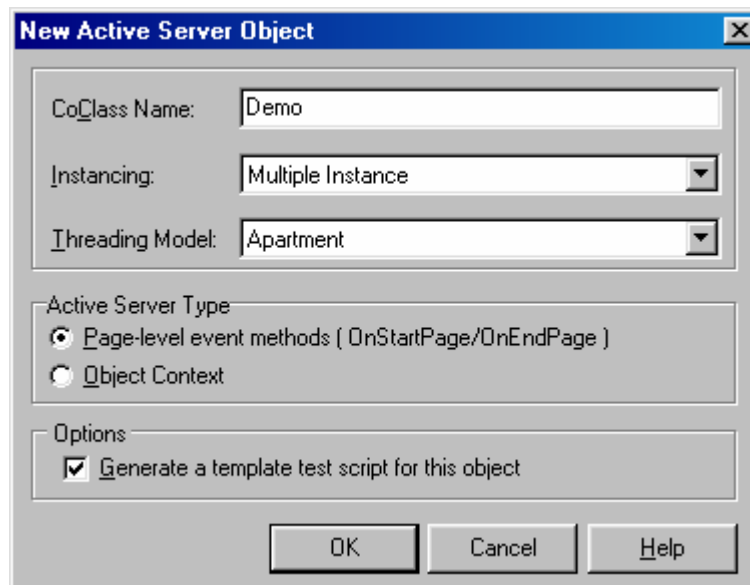
Vá ao menu "File", "New...", na paleta "Activex" escolha "Activex Library"



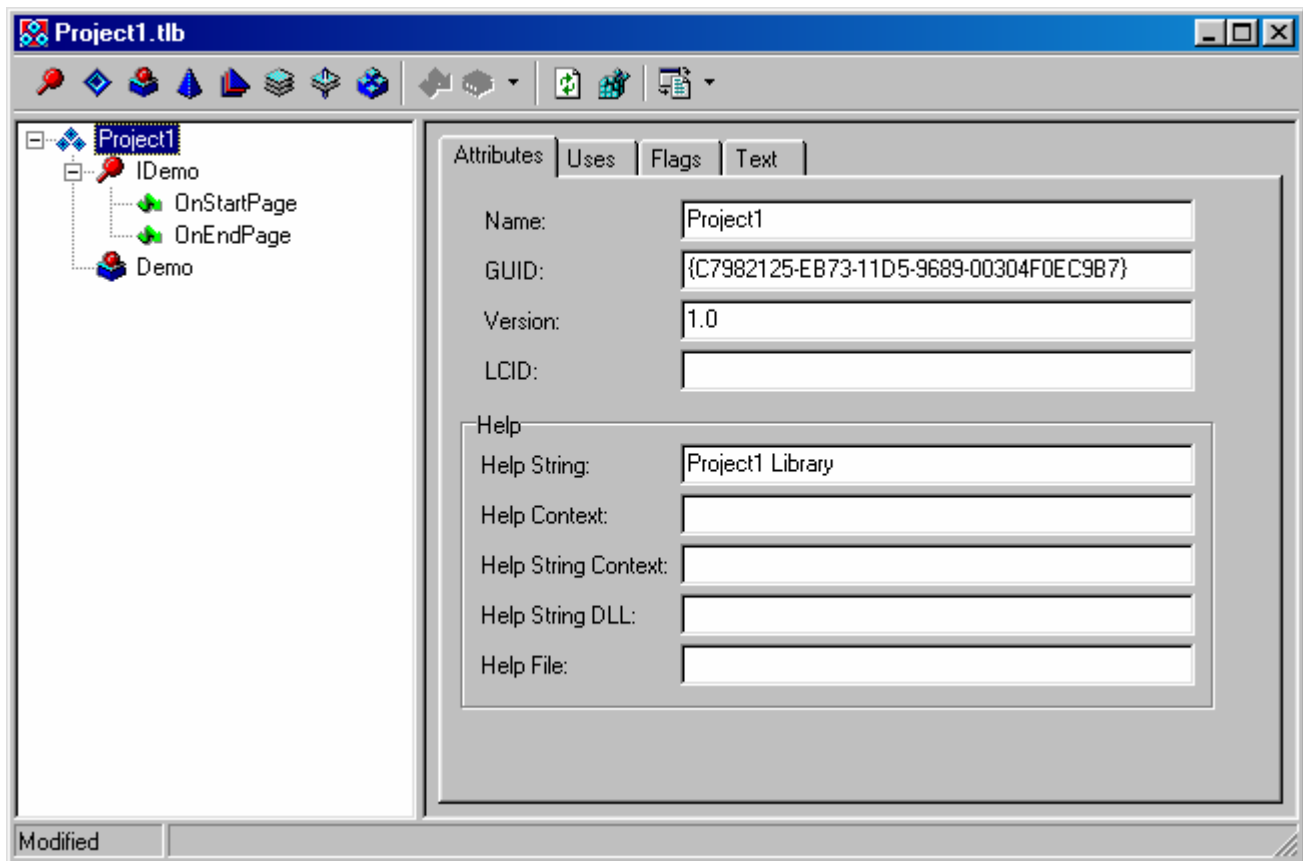
Nesse momento estamos preparando o Delphi para criar uma DLL com nosso objeto de Hello World. Você deverá receber uma tela igual a abaixo:



Volte ao menu "File", "New...", na paleta "Activex" escolha agora "Activex Server Object", aparecerá uma caixa de diálogo solicitando o nome do Objeto que você quer criar. Para fins didáticos criarei um objeto chamado Demo. Digite esse nome no campo "CoClasse Name"



Ao clicar o botão OK o Delphi apresentará o Editor da Biblioteca.



Este será nosso ambiente de criação do objeto, basicamente estaremos configurando aqui as propriedades e eventos do objeto Demo.

Antes de iniciarmos a codificação vamos salvar nosso projeto.

Clique a opção File, Save All do Menu do Delphi.

- Ao pedir o nome da Unit, troque de Unit1.pas para Exemplo1.pas.
- Delphi irá salvar também um arquivo ASP demonstrando como acionar seu componente. Salve-o com o nome de Exemplo.ASP.
- próximo diálogo solicitará o nome do projeto. Esse será o nome da DLL a ser gerada quando compilado o projeto. Mude-o de Project1.DPR para Exemplo.DPR


Perceba no Editor da Biblioteca que você terá agora uma classe chamada Exemplo com um objeto chamado Demo (A letra I que aparece na frente do nome é uma convenção indicando que trata-se de uma Interface).

O Delphi automaticamente criou dois métodos: OnStartPage e OnEndPage. Essa é uma outra convenção para objetos ASP. O Evento OnStartPage é criado todas as vezes que a página ASP for acionada, e o OnEndPage quando esta for finalizada. Nesse momento não precisaremos modificar nada nesses eventos. Finja que eles não existem.

Agora que temos um objeto chamado Demo, vamos criar nosso primeiro método: SayHello. Nosso objetivo é: todas as vezes que a página ASP acionar esse método o componente enviará para o browser do usuário a mensagem "Hello World".

Como criar o método SayHello?

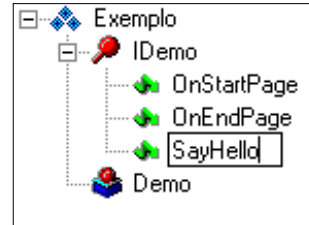
Clique sobre o Objeto Idemo selecionando-o

Clique sobre o botão  para criarmos o método SayHello

Mude então o nome do novo método para SayHello

Clique sobre o botão  para o delphi gerar O código inicial do evento.

Realizado essa etapa, já estamos prontos para Iniciar a escrita do código que enviará a Mensagem para o browser:



Pressione a tecla F12 para entrarmos no edito de código do Delphi.

Perceba que foi criado um arquivo extra de nomo Exemplo_TLB, ele é basicamente a definição da estrutura do nosso objeto. Não precisaremos mexer nele. Ignore-o. Nos interessa nesse momento é nossa unit Exemplo1. É nela que escreveremos todo o código necessário para o componente.

O Delphi já gerou toda a interface necessária para a codificação. Veja:

```
unit Exemplo1;  
  
interface  
  
uses  
    ComObj, ActiveX, AspTlb, Exemplo_TLB, StdVcl;  
  
type  
    TDemo = class(TASPObject, IDemo)  
    protected  
        procedure OnEndPage; safecall;  
        procedure OnStartPage(const AScriptingContext: IUnknown); safecall;  
        procedure SayHello; safecall;  
    end;  
  
implementation  
  
uses ComServ;  
  
procedure TDemo.OnEndPage;  
begin  
    inherited OnEndPage;  
end;  
  
procedure TDemo.OnStartPage(const AScriptingContext: IUnknown);  
begin  
    inherited OnStartPage(AScriptingContext);  
end;
```

```

procedure TDemo.SayHello;
begin

end;

initialization
  TAutoObjectFactory.Create(ComServer, TDemo, Class_Demo,
    ciMultiInstance, tmApartment);
end.

```

Vamos escrever nosso código dentro do bloco reservado para a procedure SayHello.

```

procedure TDemo.SayHello;
begin
  Response.Write('Hello World');
end;

```

Você deve estar se perguntando: **Que mágica é essa de usar um objeto do ASP (Response) dentro do Delphi?**

Não é magia nenhuma, veja na sua cláusula Uses que o Delphi fez a gentileza de adicionar a biblioteca necessária para usar as Classes do ASP dentro do seu projeto:

```

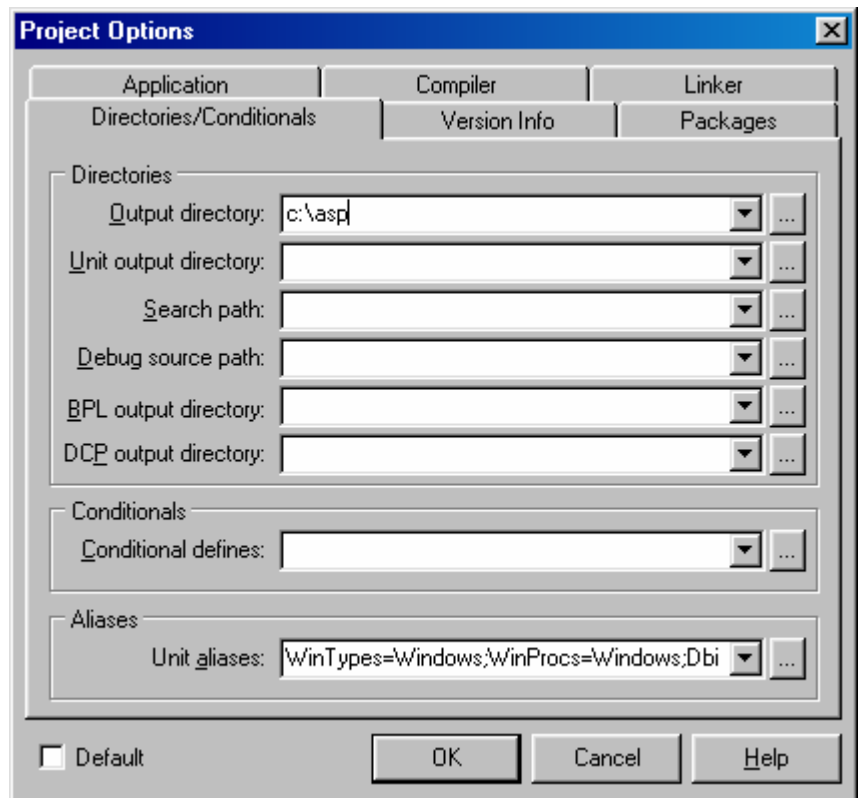
uses
  ComObj, ActiveX, AspTlb, Exemplo_TLB, StdVcl;

```

Dessa forma, você poderá utilizar de forma transparente em seu código Delphi todos os objetos "built-in" do ASP: Response, Request, Session, Server, etc. É como se você estivesse programando ASP dentro do Delphi!


Salve seu projeto e compile-o (CTRL-F9), gerando assim a DLL do componente.

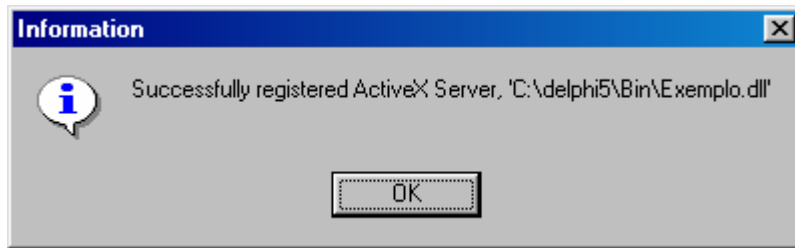
Recomendo gerar essa DLL no diretório (pasta virtual) onde a página ASP será executada. Basta salvar seu projeto nesta pasta, ou então modificar no menu "Project", "Options", "Directory Conditionals", "Output Directory" o diretório onde a DLL será criada:



Uma DLL ActiveX precisa ser registrada no Windows para poder ser instanciada pela página ASP.

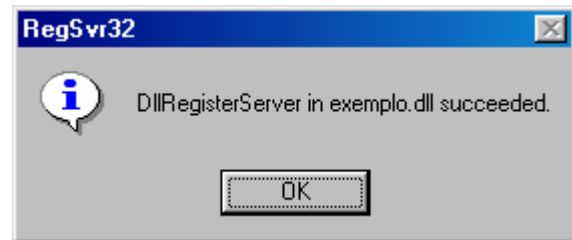
Você pode fazer isso de duas formas:

1. Clique sobre o botão  para registrar seu componente”.



2. Outra forma de se registrar o componente é: Vá ao prompt do MSDOS. Entre na pasta onde sua DLL foi gerada e digite:

```
C:\WINDOWS\SYSTEM\REGSVR32 exemplo.dll
```



Após registrar o componente precisaremos então criar a página ASP que fará sua chamada.

```
Exemplo | Exemplo.ASP | Exemplo1 | Exemplo_TLB |
<HTML>
<BODY>
<TITLE> Testing Delphi ASP </TITLE>
<CENTER>
<H3> You should see the results of your Delphi Active Server method below </H3>
</CENTER>
<HR>
<% Set DelphiASPObj = Server.CreateObject("Project1.Demo")
    DelphiASPObj.(Insert Method name here)
%>
<HR>
</BODY>
</HTML>
```

Modifique esse código para:

```
<HTML><BODY>  
<TITLE> Teste de Componente ASP </TITLE>  
<%  
    Set Obj = Server.CreateObject("Exemplo.Demo")  
    Obj.SayHello  
    Set Obj = Nothing  
%>  
</BODY></HTML>
```

Perceba na instanciação do objeto que chamamos a Classe (Exemplo) e logo em seguida o objeto (Demo).

Salve seu código ASP na pasta onde será executado.

Como executar a página ASP?

Neste tutorial estou partindo do pressuposto de que você já possui um servidor de web instalado no seu computador (PWS - Personal Web Server, ou IIS - Internet Information Server). Sem esse programa você não conseguirá executar seu ASP. Lembre-se um ASP não é uma página HTML que você pode abrir simplesmente clicando sobre ela. Ele precisa ser interpretado e para tanto é necessário a existência de um servidor WEB.

Tanto o PWS quanto o IIS podem ser obtidos gratuitamente no site da Microsoft. Se você tiver em mãos o CD do Windows 98 encontrará o PWS na pasta de Add-On's. O IIS vem com o Windows 2000 ou então no Option Pack da Microsoft.

Para executar a página EXEMPLO.ASP, abra seu browser e no endereço digite:

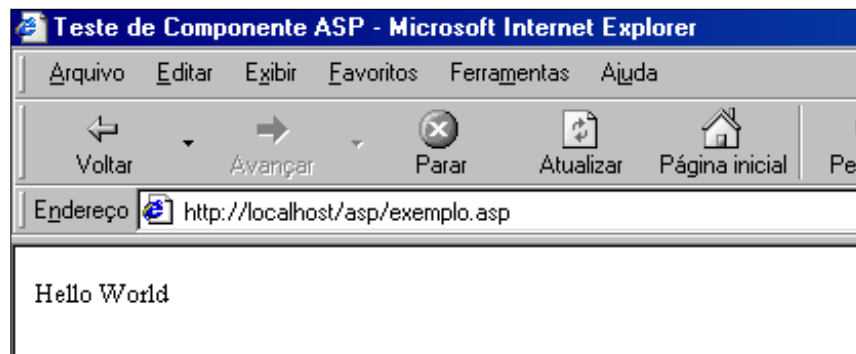
<http://localhost/asp/exemplo.asp>

O Endereço "localhost" é padrão e indica que o componente será instanciado na sua máquina local. Pode-se também usar seu número de IP local ou o nome da sua máquina na rede local em substituição ao "localhost".

Exemplos:

<http://127.0.0.1/asp/exemplo.asp> ou <http://minhamaquina/asp/exemplo.asp>

Se tudo correr bem, você verá:



Viu como é uma tarefa fácil?

Como disse anteriormente, o projeto é bastante simples, e demonstra a técnica da criação de componentes ActiveX ASP.

Implementamos apenas um método básico.

Em um próximo tutorial estarei desvendando a criação de propriedades, passagem e recepção de valores entre a página ASP e o Componente Delphi, e até mesmo tratamento de tipos mais complexos: Streams, Bitmaps, Arrays, Etc.

Apenas para finalizar:

Quando o objeto é instanciado pela página a DLL é carregada juntamente com o processo do Servidor WEB, e você não conseguirá mais recompilá-la. O Windows não permitirá o acesso à gravação do arquivo. Será apresentada uma mensagem de acesso negado na recompilação ou em qualquer tentativa de apagá-la ou renomeá-la.

Solução? Parar o servidor WEB para a descarga da DLL.

No PWS:

Vá até o diretório: c:\windows\system\inetsrv e digite:

```
PWS /STOP /Y
```

E em seguida

```
PWS /START /Y
```

Não sei porque, algumas vezes é necessário digitar o comando PWS /START /Y duas vezes para o PWS voltar a funcionar... Coisas da Microsoft.

No IIS

Você pode parar o IIS diretamente nos serviços do Windows. Basta parar o Serviço de WEB e o IIS Admin

Se quiser, pode usar o prompt do DOS:

```
NET STOP IISADMIN
```

```
NET STOP W3SVC
```

```
NET START IISADMIN
```

```
NET START W3SVC
```

Realizado essa descarga da DLL, você poderá então recompilá-la.

Esse processo não é muito prático. Depurar uma DLL será uma missão bastante estressante, pois você precisará parar e reiniciar o servidor web todas as vezes em que realizar qualquer alteração no código do componente. Em muitos casos, principalmente com o PWS, será necessário reiniciar a máquina, pois o Windows tenderá a ficar instável.

O IIS permite, mediante algumas configurações especiais, a execução da DLL sem carregá-la no mesmo processo do servidor web, permitindo inclusive depurar o componente dentro do delphi ao mesmo tempo em que o ASP está sendo executado. Num próximo tutorial ensinarei como realizar essa configuração.

Outros tutoriais:

- Como criar Componentes ASP em Visual Basic
- Como Instalar e Configurar o PWS
- Como Debugar componentes ASP no Delphi com IIS
- Explorando a criação de Componentes ASP em Delphi

Comunicando-se com o objeto através de parâmetros

Esta seção na fazia parte deste tutorial, no entanto resolvi incorporá-lo para os desenvolvedores que desejarem se aprofundar na técnica de criação de componentes.

Serão alguns exemplos que mostrarão como passar parâmetros para o objeto e receber um resultado processado pelos métodos internos.

Será desenvolvido um componente com 2 propriedades de escrita, uma de leitura e um método.

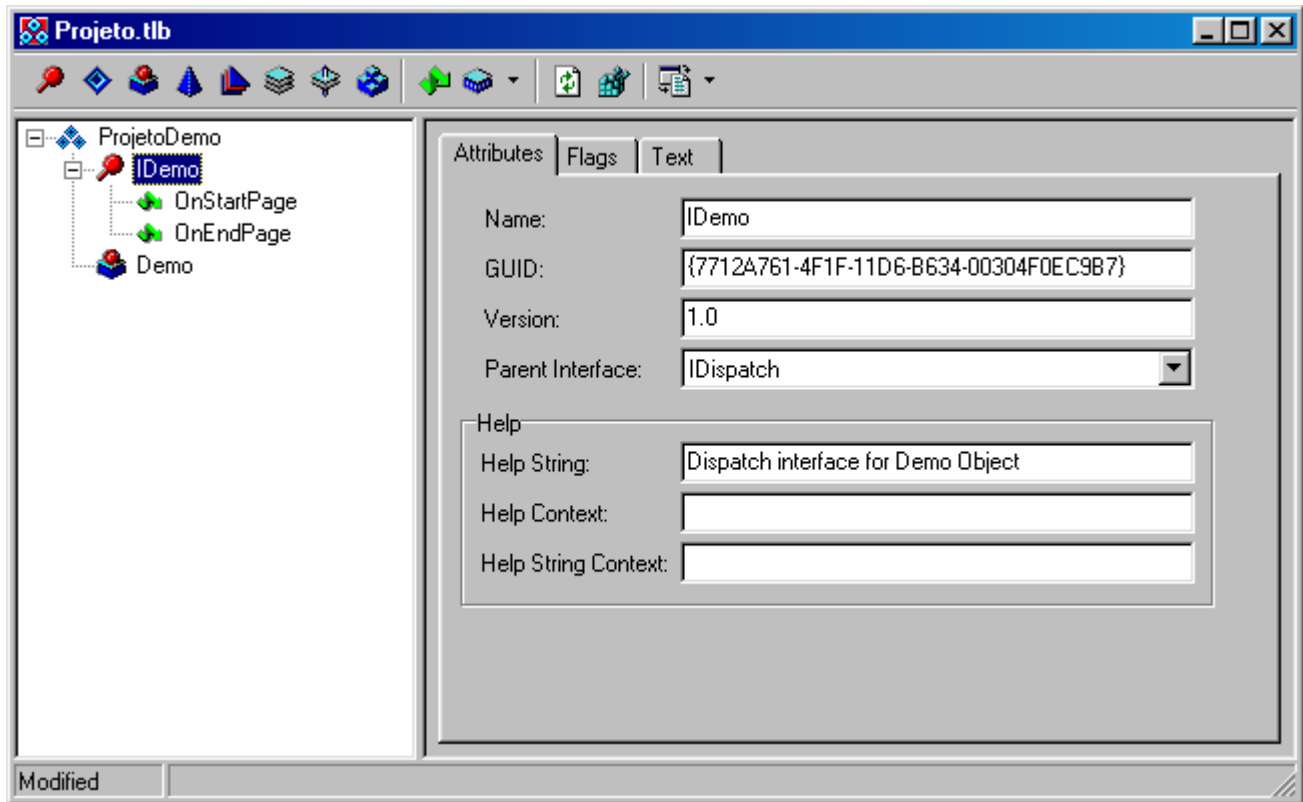
A função do objeto será receber através das propriedades de escrita um Valor Unitário e uma Quantidade; quando executado o método CalculaTotal, o valor total (resultado da multiplicação do Valor Unitário pela Quantidade) será retornado na propriedade "só de leitura" Valor Total.

A estrutura de procedimentos será:

1. Guarda-se o Valor Unitário: `obj.ValorUnitario = 10`
2. Guarda-se a Quantidade: `obj.Quantidade = 2`
3. Executa-se o cálculo: `obj.CalculaTotal`
4. Recupera-se o resultado do cálculo: `valor = obj.ValorTotal`

Serão apresentadas 3 formas distintas de se calcular o Valor Total (via propriedades, via parametros e via variável por referência).

Comunicando-se com o objeto através de parâmetros

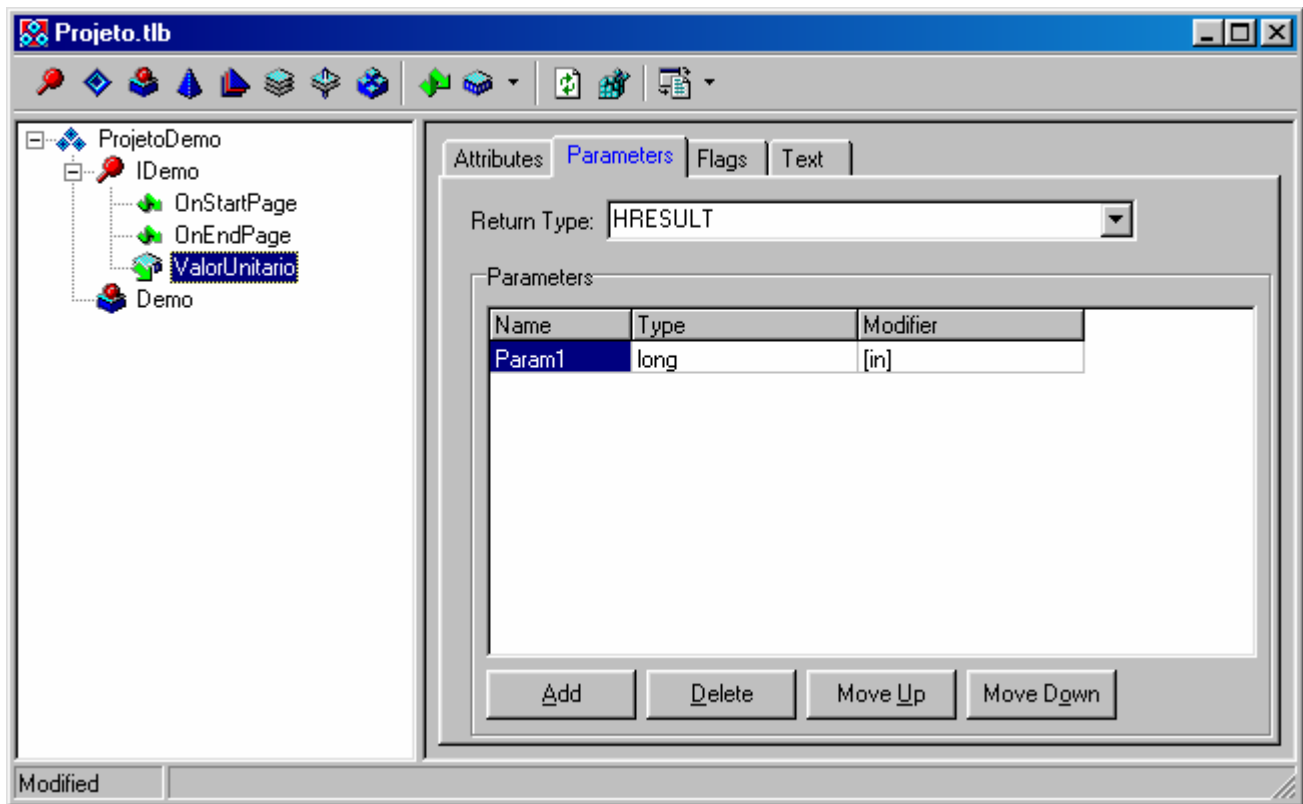


Os métodos OnStartPage e OnEndPage são necessários para a instanciação do objeto dentro de páginas ASP.

Para instanciar o objeto em uma pagina ASP a sintaxe é:

```
Set Obj = Server.CreateObject("ProjetoDemo.Demo")
```

Definindo a Propriedade que receberá o conteúdo do Valor Unitário:



Esta é uma propriedade só de escrita e servirá para enviar ao objeto o Valor Unitário.

Exemplo:

```
Obj.ValorUnitario = 100
```

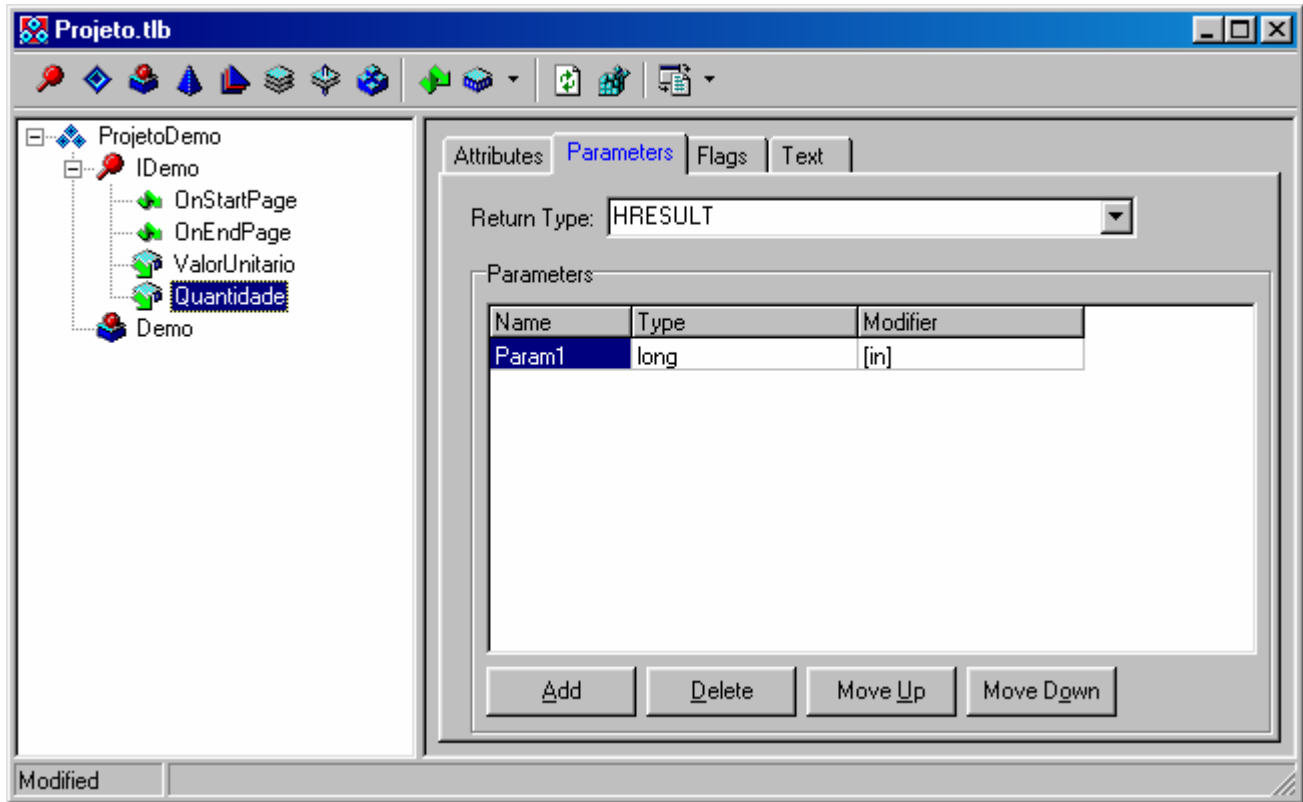
O código interno da procedure que ajustará o valor será:

```
procedure TDemo.Set_ValorUnitario(Value: Integer);  
begin  
    FValorUnitario := Value;  
end;
```

Obs: Não esqueça de definir na seção Interface da Unit do Objeto as variáveis:

```
FvalorTotal, FvalorUnitario, Fquantidade : Integer (ou OleVariant);
```

Definindo a Propriedade que receberá o conteúdo de Quantidade:



Esta é uma propriedade só de escrita e servirá para enviar ao objeto a Quantidade.

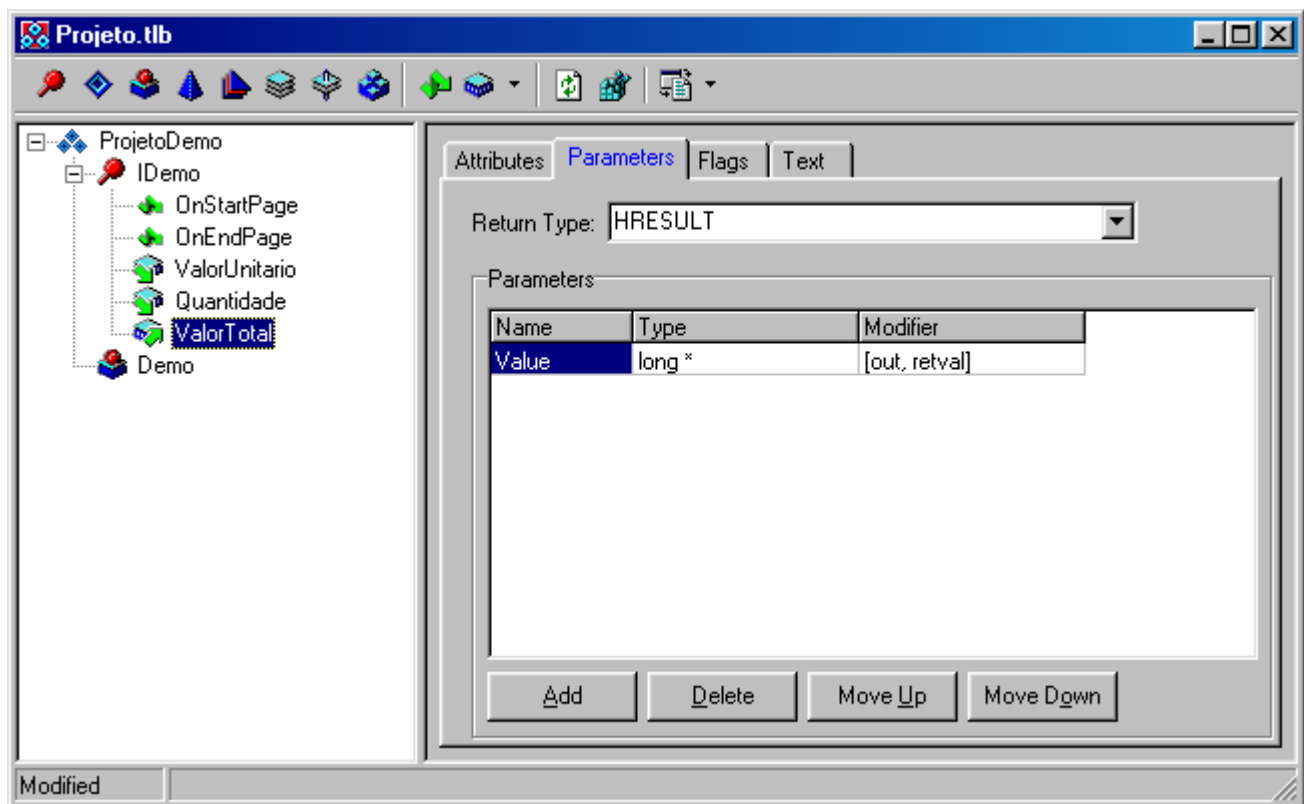
Exemplo:

```
Obj.Quantidade = 2
```

O código interno da procedure que ajustará a Quantidade será:

```
procedure TDemo.Set_Quantidade(Value: Integer);  
begin  
    FQuantidade := Value;  
end;
```

Definindo a Propriedade que retornará o Valor Total:



Esta é uma propriedade só de leitura e servirá para retornar do objeto o resultado obtido no cálculo da Quantidade * ValorUnitário.

O resultado só estará disponível após a execução do método apropriado de cálculo.

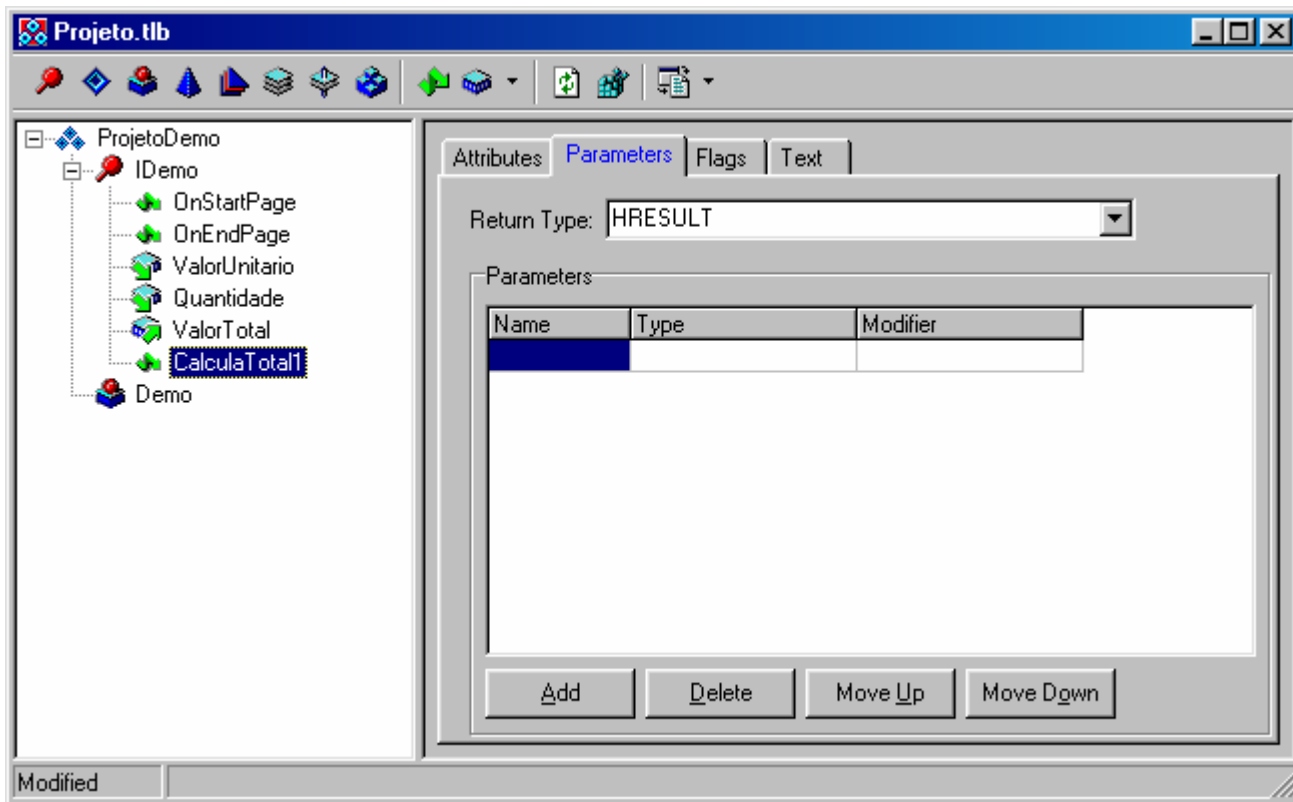
Exemplo:

```
Total = Obj.ValorTotal
```

É necessário que a definição do parâmetro contenha o "Modifier" em **[out,retval]**, por se tratar de uma função que retorna um valor ao chamador. O código interno será:

```
function TDemo.Get_ValorTotal: Integer;  
begin  
    Result := FValorTotal;  
end;
```

Definindo o Método que calculará o Valor Total (1ª forma):



Esta primeira modalidade de cálculo do Valor Total usará a técnica de recepção das informações de origem e retorno através de propriedades do objeto.

Dessa forma, a Quantidade e o Valor Unitário serão passados através das respectivas Propriedades do objeto, será então aplicado o método CalculaTotal1 (sem parâmetros) e o resultado será obtido através da leitura da propriedade ValorTotal do Objeto.

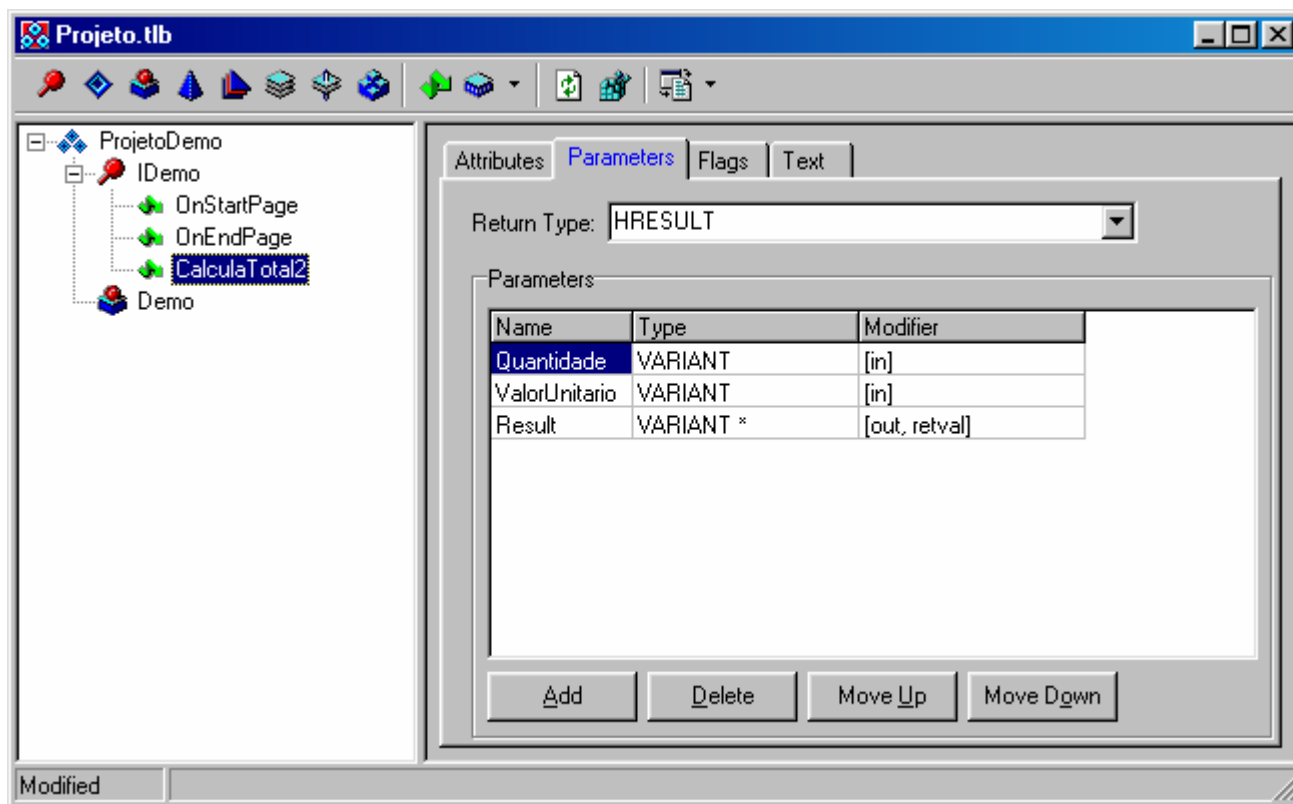
Exemplo:

```
Obj.Quantidade = 2  
Obj.ValorUnitario = 10  
Obj.CalculaTotal1
```

O método CalculaTotal1 executará o cálculo e armazenará o resultado na propriedade ValorTotal do objeto. O código interno será:

```
procedure TDemo.CalculaTotal1;  
begin  
    FValorTotal := FQuantidade * FValorUnitario;  
end;
```

Definindo o Método que calculará o Valor Total (2ª forma):



Este método não utiliza as propriedades Quantidade e ValorUnitario para enviar os valores ao objeto, ele utiliza parâmetros para passá-los e obtém o cálculo como resultado da chamada.

Exemplo:

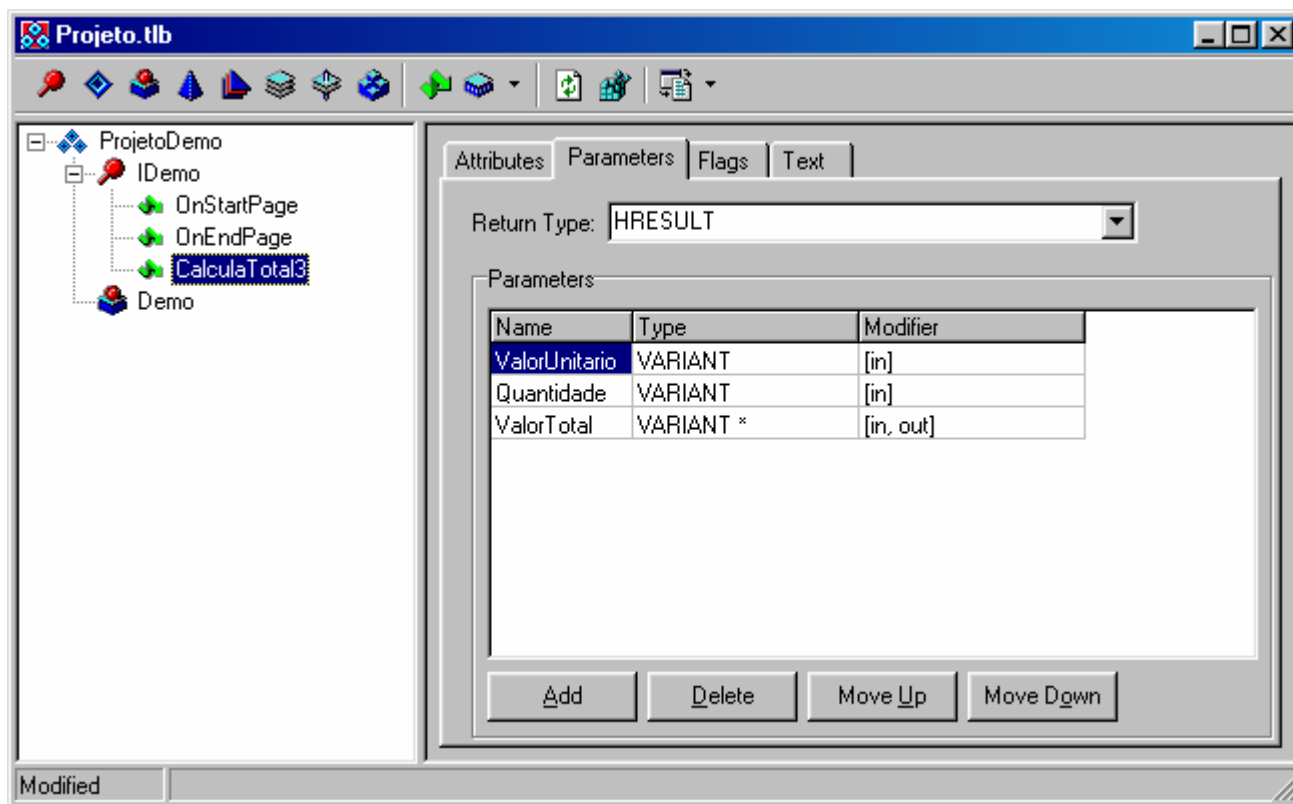
```
Vr = Obj.CalculaTotal2(2,10)
```

Ao ser executado, o método realizará o cálculo e retornará o valor na variável VR. O primeiro parametro é a Quantidade e o segundo o Valor Unitário. O código interno será:

```
function TDemo.CalculaTotal2(Quantidade, ValorUnitario: OleVariant): OleVariant;  
begin  
    Result := Quantidade * ValorUnitario;  
end;
```

Obs.: É altamente recomendado o uso de tipos variantes em objetos ActiveX, visando evitar incompatibilidades de tipos com outras linguagens.

Definindo o Método que calculará o Valor Total (3ª forma):



Esta não é uma forma muito comum de passagem de parâmetros, contudo é uma alternativa interessante. O método será invocado com 3 parâmetros, sendo os dois primeiros passados por valor (contendo a quantidade e o valor unitário) e o terceiro por referência, servirá como recipiente de retorno do Valor Total.

Exemplo de uso:

```
Vr = 0  
Obj.CalculaTotal3(2,10,Vr)
```

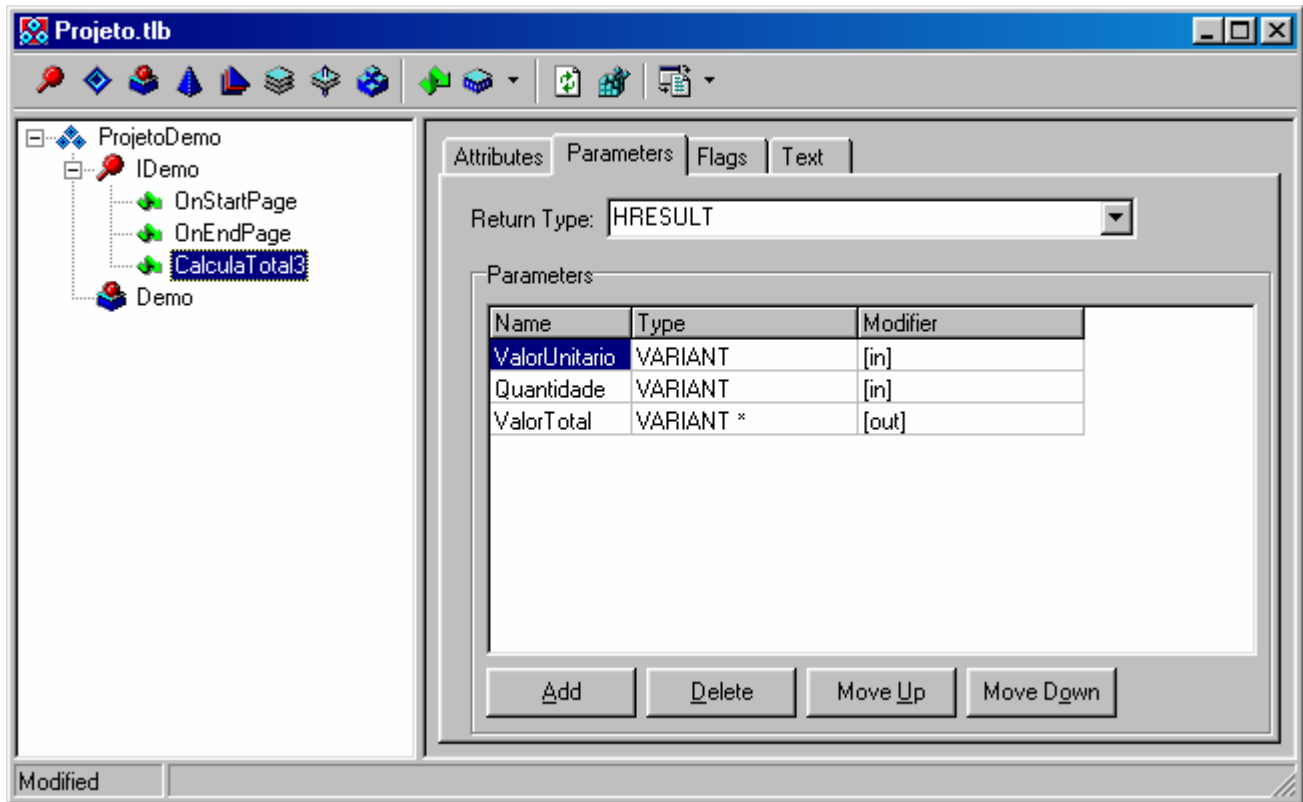
Após a execução a variável VR conterà o resultado do cálculo.

Código Interno:

```
procedure TDemo.CalculaTotal3(ValorUnitario, Quantidade: OleVariant; var ValorTotal:  
OleVariant);  
begin  
    ValorTotal := Quantidade * ValorUnitario;  
end;
```

Veja que os "Modifiers" do parâmetro de retorno (valor total) foram utilizadas as opções [in,out] dessa forma, qualquer valor passado na variável ValorTotal pelo programa chamador poderá ser utilizado pelo objeto antes da realização do cálculo do valor total.

Uma outra alternativa seria retirando o "Modifier" [in] e deixando apenas [out], neste caso a variável ValorTotal passada como referência será zerada ao iniciar a execução do código do método. A Type Library terá a seguinte aparência:



O código gerado será:

```
procedure TDemo.CalculaTotal3(ValorUnitario, Quantidade: OleVariant; out ValorTotal: OleVariant);  
begin  
    ValorTotal := Quantidade * ValorUnitario;  
end;
```