

Nesta seção você encontra artigos para iniciantes em .net

XML Web Services

Desenvolvendo e consumindo com .NET

Neste artigo veremos

Introdução aos XML Web Services; Criação de um Web Service usando o Visual Studio .NET 2005; Consumo do Web Service através de uma aplicação Windows Forms; Consumo do Web Service através de uma aplicação ASP.NET.

Qual a finalidade

Este artigo procura facilitar os primeiros passos de quem deseja começar a desenvolver e consumir Web Services.

Quais situações utilizam esses recursos?

Em serviços de informação para consultar preços de produtos, cotações ou comparações de preço, horários de vôos ou ainda

serviços de tradução. Estas informações podem ser acessadas através de páginas da Internet, documentos, aplicações Windows ou mesmo por dispositivos móveis; Os Web Services também podem ser utilizados como repositório de dados possibilitando que dispositivos e softwares de diversas plataformas possam acessar estes. Serviços como agendas baseadas na Internet e armazenamento de arquivos são beneficiados com a utilização dos Web Services; Integração de dados e análise. Os Web Services podem ser usados, por exemplo, para que uma empresa de transportes receba solicitações sobre uma viagem que deva ser feita e indique através da análise dos dados qual a melhor rota.



Vladimir Rech

vladimirrech@yahoo.com.br

Tecnólogo em Desenvolvimento de Sistemas pelo CEFET-PR, palestrante; trabalha com desenvolvimento de sistemas em .NET destacando-se aplicações Windows, ASP e Web Services.



Resumo do DevMan

Os Web Services são um tipo especial de software que fornecem funcionalidades através de uma rede ou mesmo da Internet. Com eles é possível que se tenha um processamento distribuído, onde computadores separados fisicamente realizem tarefas distintas. Através destes é possível ter a lógica de uma aplicação em computadores separados fisicamente e permitir que outras máquinas façam uso desta lógica sem necessitar de programas mais complexos

Com o crescimento da importância da Internet, muito cedo surge a necessidade de se usar não somente os dados localizados em um servidor distante, como também consumir a lógica e as regras de processamento existentes neste.

Numa aplicação típica cliente-servidor, um *software* no computador local, busca os dados em um servidor, que geralmente possui um gerenciador de banco de dados, realiza o processamento localmente e envia estes dados de volta para o servidor.

Os *Web Services* permitem que o armazenamento, agregação e análise dos dados seja feito em um servidor e páginas *HTML*, aplicações *Windows* ou mesmo dispositivos móveis como *Pockets PC's*, *Smartphones*.

Neste artigo, será feita uma introdução aos *Web Services* desenvolvendo-os usando *Microsoft .NET* e o *Visual Studio 2005*. Veremos como criar um exemplo simples e como aplicações *ASP.NET* e *Windows Forms* podem consumir estes.

Embora seja um exemplo bem simples é o suficiente para termos uma noção de como criar Web Services, pois a estrutura usada para outros casos é a mesma.

Criando o Web Service

No nosso exemplo, criaremos um *Web Service* que irá receber como parâmetros de entrada dois números para que seja realizada uma operação aritmética, um parâmetro do tipo *string* indicando qual a operação desejada: soma, subtração, multiplicação ou divisão, devolvendo o resultado da operação.

Embora seja um exemplo bem simples é o suficiente para termos uma noção de como criar Web Services, pois a estrutura usada para outros casos é a mesma

Abra o *Visual Studio* e selecione o menu *File -> New -> Project*. Na janela da **Figura 1** escolha *ASP.NET Web Service Application*. Como linguagem vamos usar *C#*, embora seja possível criar

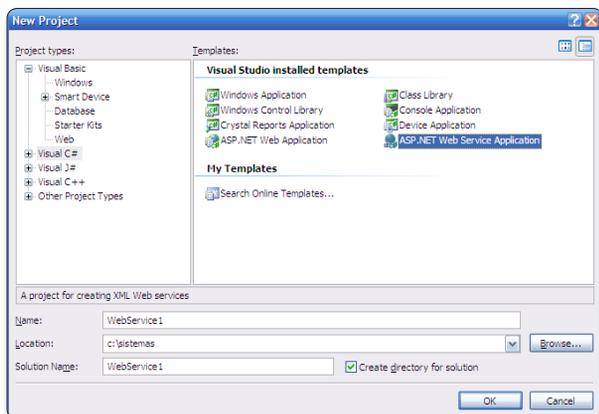


Figura 1. Janela com as opções para criação do novo projeto

em outras linguagens como *Visual Basic .NET*. Selecione o tipo da linguagem no campo *Project Types*.

No campo *Name* digite um nome para seu projeto como, por exemplo, *WebService1*. Escolha uma pasta para que o projeto seja criado, no nosso caso, será usada a pasta *c:\sistemas*. O *Visual Studio* criará uma pasta com o nome do projeto com toda a estrutura necessária automaticamente.

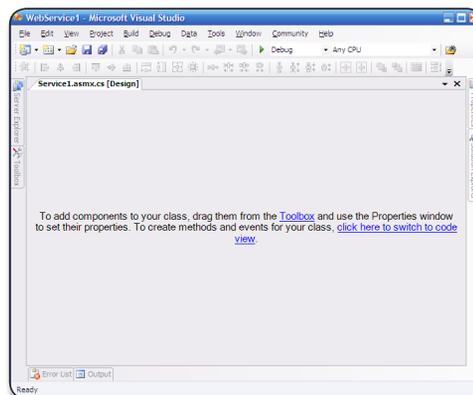


Figura 2. Janela inicial do projeto

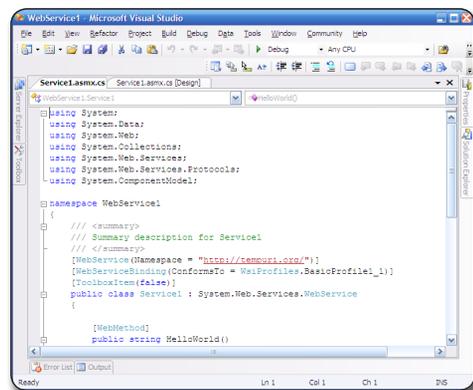


Figura 3. Código-fonte em C#

Esta pasta será somente para uso durante o desenvolvimento. Os *Web Services* são publicados no servidor de Internet como o *Internet Information Service (IIS)*. Posteriormente, veremos como



Nota do DevMan

Web Services não são uma exclusividade de uma plataforma como o *Windows* ou *.NET*, em vez disto, seguem padrões estabelecidos internacionalmente e podem ser implementados em um grande número de plataformas e linguagens.

No caso do *Visual Studio*, sua implementação permite uma produtividade grande graças ao fato de que a ferramenta facilita o trabalho ocultando vários detalhes da implementação através da geração automática de código.

No entanto, sua relevância está justamente no fato de os *Web Services* utilizarem padrões aceitos no mundo inteiro, e homologado por entidades referência. Um *Web Service* criado em *Java* deve ser capaz de ser consumido normalmente por uma aplicação feita em *.Net*, e vice-versa.

publicar o *Web Service* neste servidor. Após nomear o projeto e escolher a localização do mesmo clique em OK.

Este projeto está sendo desenvolvido usando-se o Windows XP com o IIS instalado. Pode-se desenvolver em qualquer versão do Windows que suporte o .NET 2.0. A maior parte dos servidores atuais roda Windows 2003 com o IIS 6.

Listagem 1. Código que executa a operação

```
[WebMethod]
public decimal OpArit(decimal n1, decimal n2, string op, out
string erro)
{
    // Todo parâmetro de saída (out) deve ser obrigatoriamente
    inicializado.
    erro = "";
    decimal ret = 0;
    // Verifica qual a operação a ser executada
    switch (op)
    {
        case "+": ret = n1 + n2; break;
        case "-": ret = n1 - n2; break;
        case "x": ret = n1 * n2; break;
        case "/": if (n2 > 0)
            {
                ret = n1 / n2;
            }
            else
            {
                erro = "Divisão por zero.";
            } break;
        default: erro = "Nenhum operador válido."; break;
    }
    return ret;
}
```

Usualmente será exibida a tela da **Figura 2**, mas normalmente trabalhamos com o código-fonte em C#. Para exibir este, clique no link *click here to switch to code view* para exibir o código como na **Figura 3**. Se você estiver utilizando o *Visual Studio 2008*, ele já exibirá a janela de código, por padrão.

O código da **Listagem 1** exibe a operação usada, e está organizada da seguinte forma: os parâmetros *n1* e *n2* são os números que devem ser usados na operação. O parâmetro *op* é a operação que deve ser executada.

Observe o parâmetro *erro* em *out string erro*. Este parâmetro é de saída, ou seja, retorna para a aplicação que chamou o *Web*

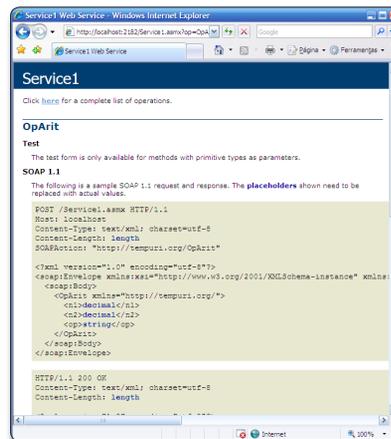


Figura 5. Detalhes do serviço OpArit do Web Service

Listagem 2. Código para testar o Web Service a partir do Visual Studio

```
[WebMethod]
public string teste(decimal n1, decimal n2, string op)
{
    string erro = "";
    decimal ret = OpArit(n1, n2, op, out erro);

    if (erro != "")
    {
        return erro;
    }
    return ret.ToString();
}
```

Service os erros capturados durante a operação.

O bloco *switch* verifica qual o operador e executa o cálculo. Destacando que na operação de divisão, indicada pelo operador */*, é feito um teste se o divisor é zero uma vez que não é possível haver divisão por zero para evitar erros na aplicação. Neste mesmo bloco é feito um teste caso nenhum operador válido seja informado.

Para um método ser visível para aplicações externas ao *Web Service*, deve iniciar com a declaração `[WebMethod]` e ser `public`. Você pode ter funções internas que não sejam disponíveis para consumo. Basta omitir a declaração `[WebMethod]`. Além disso, o nome dos métodos não pode ter o mesmo nome da classe.

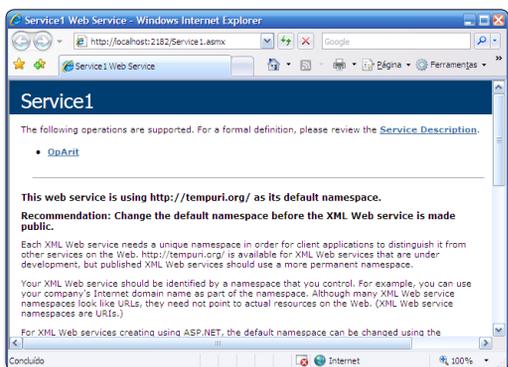


Figura 4. Página de teste do Web Service

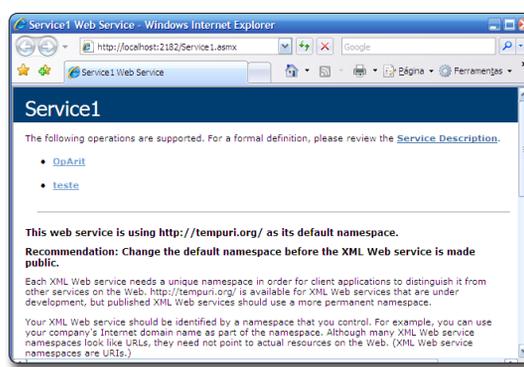


Figura 6. Declaração das operações do Web Service agora com a operação teste

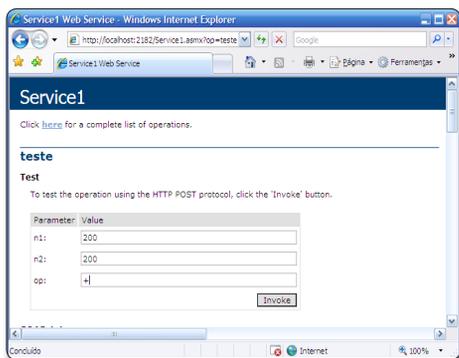


Figura 7. Serviço teste sendo testado on-line

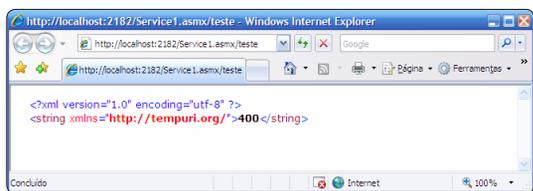


Figura 8. Resultado da operação da Figura 7, note o valor retornado em XML

Normalmente, ao ser executado na máquina local, um Web Service permite que os parâmetros sejam digitados e o mesmo seja testado. Neste exemplo, como temos o parâmetro de saída out string erro, não é possível fazer o teste.

Caso seja necessário um teste antes de desenvolver a aplicação consumidora do Web Service, pode-se desenvolver um método público para isso, como na Listagem 2.

Executando testes

O Visual Studio permite que se teste o Web Service a partir de seu IDE, bastando para isso pressionar a tecla F5. Se tudo estiver certo a tela da Figura 4 é exibida. Se clicarmos em OpArit uma descrição detalhada do mesmo é exibida como exibido na Figura 5.

Agora, pressionando-se F5 é possível testar o Web Service. As Figuras 6 e 7 mostram as telas de teste e a Figura 8 o resultado do teste.

Publicando o Web Service

Para que outras aplicações possam usar o Web Service é necessário publicá-lo. Para isso, vamos usar o gerenciador de sites da Internet IIS versão 6. No nosso caso, darei o exemplo usando o Windows XP, mas, é possível fazer com outras versões, como o Windows Vista, mas principalmente Windows Server 2003 e 2008 que são sistemas operacionais para servidores, mais indicados para esta função.

Antes disto, devemos criar o diretório onde a aplicação ficará armazenada e isso pode ser feito a partir do próprio Visual Studio. No Solution Explorer, clique com o botão direito do mouse sobre o projeto do Web Service, no menu que se abre escolha Publish, como na Figura 9.

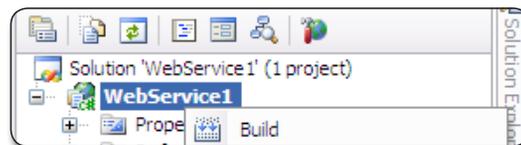


Figura 9. Publicando o Web Service com o Visual Studio

Informe a pasta onde deseja manter a aplicação funcionando e selecione as opções como na Figura 10.

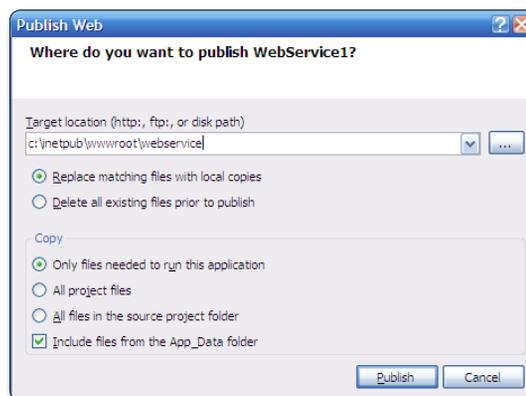


Figura 10. Configurando a pasta para a publicação do Web Service

O passo seguinte é configurar o IIS para gerenciar o acesso à aplicação. Abra o Painel de Controle do Windows, selecione Ferramentas Administrativas, e em seguida, clique no ícone Internet Information Services, como exibido na Figura 11.

É possível usar qualquer pasta para publicar o Web Service, embora, neste caso esteja sendo usada a pasta padrão do IIS – c:\inetpub\wwwroot.

Também, como indicado na janela de diálogo Target location (http:, ftp:, or disk path), pode-se publicar a aplicação em computadores remotos, facilitando bastante esta tarefa.

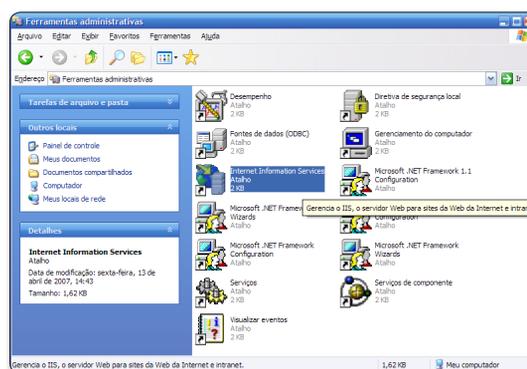


Figura 11. Acessando o IIS

Para criar a aplicação expanda o ícone Internet Information Services, em seguida faça o mesmo com o ícone correspondente ao nome do computador local em que se está publicando o Web Service, expanda também Sites da Web e por fim, com o

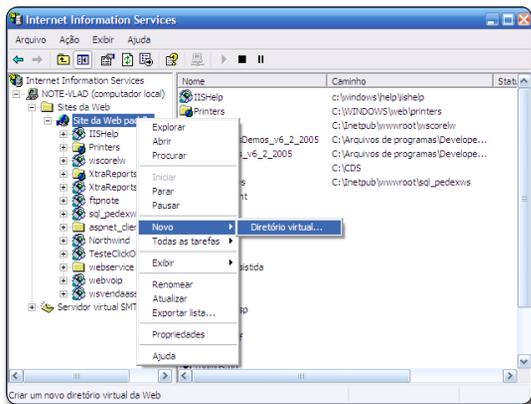


Figura 12. Criando uma nova aplicação

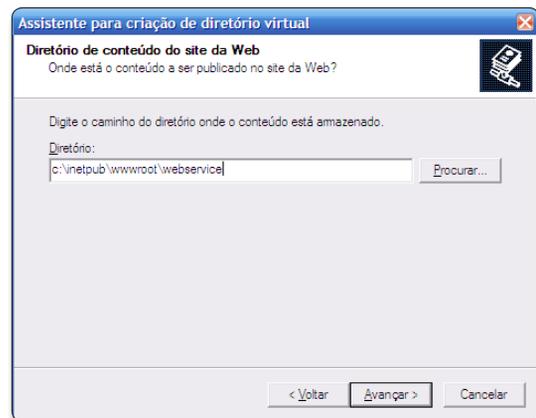


Figura 15. Seleção do diretório

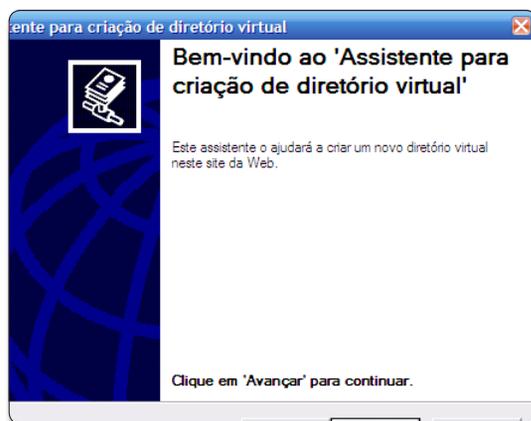


Figura 13. Assistente do IIS para publicação do Web Service

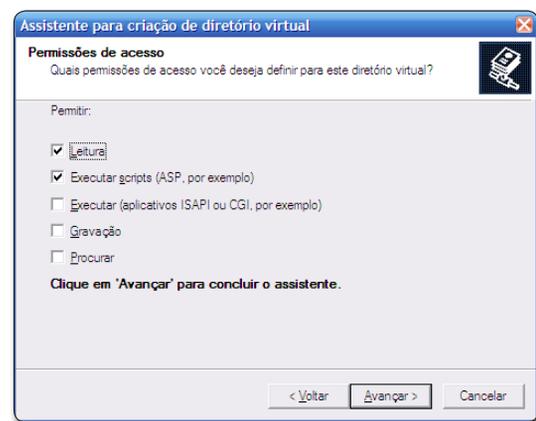


Figura 16. Configuração das permissões de acesso

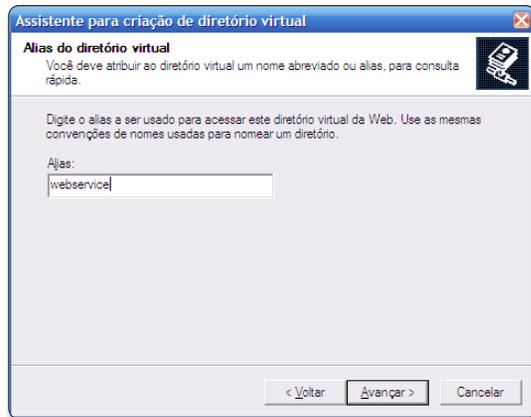


Figura 14. Seleção do nome do Web Service para publicação

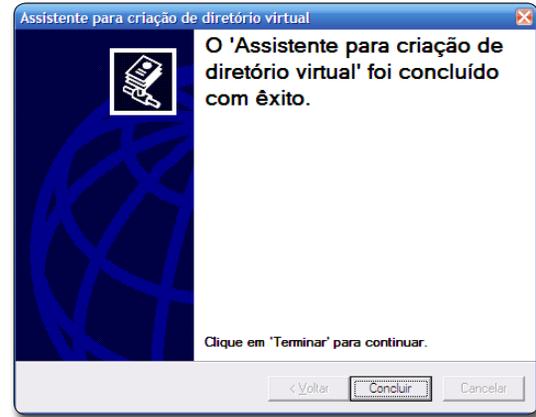


Figura 17. Conclusão da configuração

botão direito do mouse clique em *Site da Web Padrão* e escolha *Novo -> Diretório virtual*, como exibido na Figura 12.

Configure o local da aplicação usando o assistente do IIS conforme é demonstrado nas Figuras 13, 14, 15, 16 e 17.

Um detalhe importante: caso haja duas versões do *Microsoft Framework .NET* instalado no computador onde o *Web Service* será publicado, é necessário configurar qual a versão do mesmo será usada, no nosso caso a 2.0. Isto é necessário para que a aplicação seja executada corretamente. Para configurar, selecione o *Web Service* recém criado, ainda no IIS, clique com o botão direito do

mouse e escolha a opção *propriedades*. Na janela que se abre, selecione a aba *ASP.NET* e configure o campo *ASP.NET version* para a versão apropriada (2.0.50727) como na Figura 18.

Consumindo um Web Service através de uma aplicação Windows Forms

O propósito dos *Web Services* é prover uma lógica para aplicações remotas. Uma forma de consumir estes *Web Services* é através de programas *Windows Forms*. A seguir, será demonstrado como fazer isto.

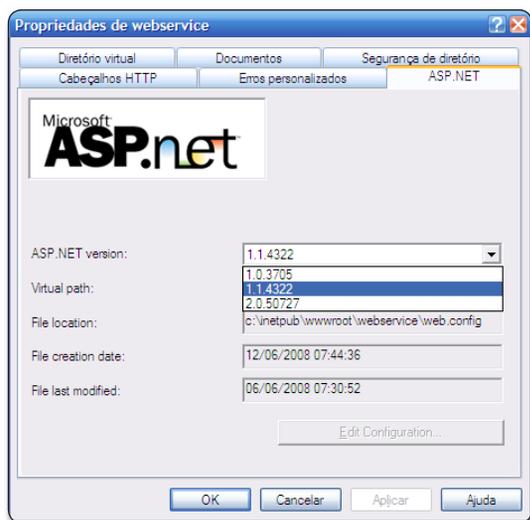


Figura 18. Configurando a versão do Framework

Abra o *Visual Studio* e carregue a *Solution* criada para o *Web Service*. Iremos criar um novo projeto nesta mesma *Solution*.

Embora, estejamos usando a mesma *solution* para o *Web Service* e as aplicações consumidoras, isto não é necessário, podendo-se criar a outra aplicação em outra *solution*.

No *Solution Explorer*, sobre o ícone da *Solution*, clique com o botão direito do mouse e escolha *Add New Project*, conforme na Figura 19.

Para o nosso novo projeto escolha como *template* o item *Win-*

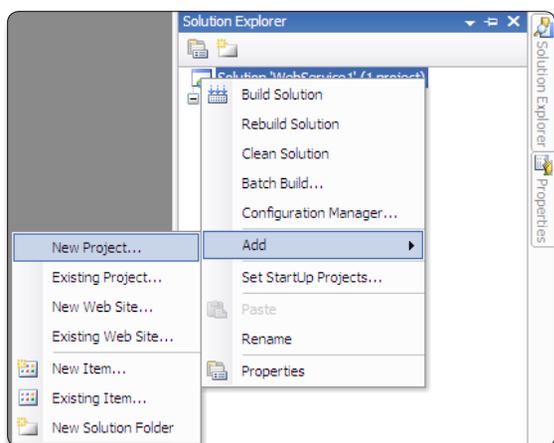


Figura 19. Adicionando um novo projeto à solution

ows Application se não estiver selecionado, selecione em *Project Types* a opção *Visual C#* para utilizarmos o *C#* como linguagem da nossa aplicação. Veja a Figura 20 para tirar dúvidas.

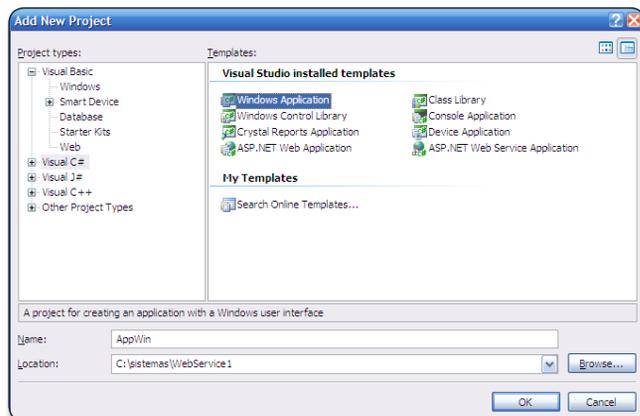


Figura 20. Selecionando o tipo do novo projeto

Um formulário é criado por padrão: *Form1.cs* e é exibido. No *Form Designer* que foi aberto, arraste os controles como na Figura 21 e configure suas propriedades da seguinte forma:

- **Formulário:**
 - (*Name*): *frmPrincipal*;
 - *FormBorderStyle*: *FixedDialog*;
 - *Text*: Invocando Web Service
- **Label 1**
 - *Font*: *Arial*; *15,75pt*; *style=Bold*
 - *Text*: Invocando Web Service
- **Label 2**
 - *Text*: Número 1
- **Text Box 1**
 - (*Name*): *txtNumero1*
- **Label 3**
 - *Text*: Número 2
- **Text Box 2**
 - (*Name*): *txtNumero2*
- **Label 4**
 - *Text*: Operação
- **Combo Box:**
 - (*Name*): *cmbOperacao*
 - *DropDownStyle*: *DrowpDownList*
 - *Items*: Soma; Subtração; Multiplicação; Divisão
- **Label 5:**
 - *Text*: Resultado
- **Text Box 3:**
 - (*Name*): *txtResultado*
 - *ReadOnly*: *True*
- **Buttom:**
 - (*Name*): *btnExecutar*

O próximo passo é fazer a referência ao *Web Service* para que sejam criadas automaticamente pelo *Visual Studio* as classes de acesso ao *Web Service*. Para fazer isso, clique com o botão direito sobre o ícone do projeto em *Solution Explorer* e escolha *Add Web Reference* como na Figura 22.

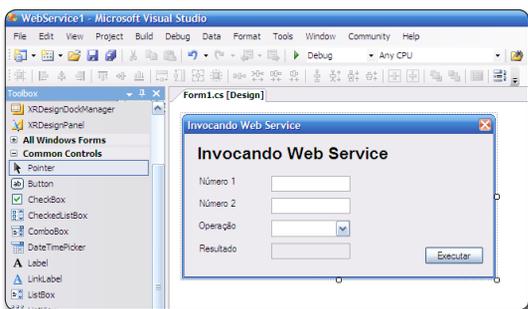


Figura 21. Form Designer

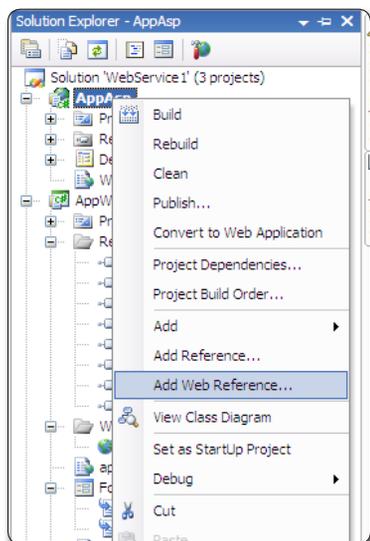


Figura 22. Como adicionar uma Web Reference

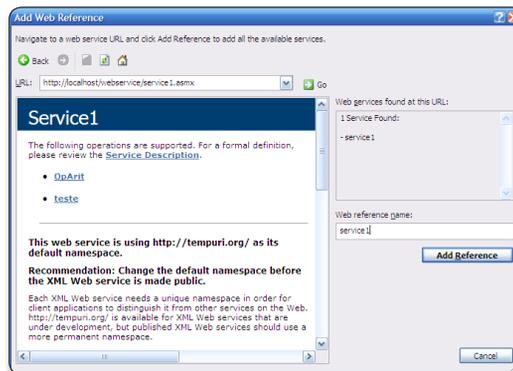


Figura 23. Adicionando a referência para acessar o Web Service

Destacam-se no código o tratamento de erros tanto na conversão dos números digitados como na execução do *Web Service*. Note também que estamos configurando o endereço do *Web Service* manualmente (*servico.Url*) mas podemos buscar esta informação em um arquivo de configuração por exemplo. Uma vez declarado o objeto para acesso ao *Web Service* o método desejado é acessado como em qualquer classe. Se tudo estiver OK, execute o código pressionando F5 e execute testes. Veja na Figura 24 um exemplo da aplicação sendo executada.

Vale a pena lembrar: aqui, estamos consumindo um *Web Service* desenvolvido em C# em uma outra aplicação desenvolvida na mesma linguagem. Porém, nada impede que aplicações desenvolvidas em outras linguagens e plataformas acessem o nosso *Web Service*.

Consumindo Web Services a partir de uma aplicação Web

Da mesma forma que fizemos com aplicações *Windows*, o uso dos *Web Services* a partir de uma aplicação *ASP.NET* é muito fácil usando o *Visual Studio*. Os passos são basicamente os mesmos sendo a principal diferença o uso da interface usando *tags HTML*.

Abra o *Visual Studio* e carregue novamente a *Solution* na qual foi criado o *Web Service*. No ícone da *Solution* clique com o botão direito do mouse e escolha *Add Project* como na Figura 19. Na janela que se abre, selecione em *Project Types*, *Visual C#* e em *Templates*, *ASP.NET Web Application* como na Figura 25. Nomeie a aplicação como *AppAsp* (ou um nome que achar apropriado) e clique em OK. Inclua os elementos da página como na Figura 26. O código *HTML* está detalhado na Listagem 4.

Após configurar o *design*, vamos criar a referência para o *Web Service*. Para isto, selecione o projeto *ASP* na *Solution Explorer*, clique com o botão direito do mouse e escolha *Add Web Reference*. No campo *Web reference name* da janela exibida, digite o nome que será usado para a classe a ser criada automaticamente pelo *Visual Studio* e que deverá ser usado para referenciar o serviço dentro da aplicação. Clique em *Add Reference* para que a operação seja completada. Observe que na

O passo seguinte é indicar na janela qual o endereço do *Web Service* que se pretende consumir. Como iremos acessar um *Web Service* localizado no próprio computador, no campo *URL* digite *http://localhost/webservice/service1.aspx* (Figura 23). Se nosso *Web Service* estivesse localizado externamente usaríamos um endereço apropriado onde trocaríamos o *localhost* pelo nome ou *ip* do servidor, *webservice* pelo nome dado ao *Web Service* no servidor e *service1.aspx* pelo nome do serviço a ser acessado. Após isso, clique no botão *Go*. Se tudo estiver correto, será apresentada a página do *Web Service* como na Figura 23.

No campo *Web reference name* digite o nome que será usado para a classe a ser criada automaticamente pelo *Visual Studio* e que deverá ser usada para referenciar o serviço dentro da aplicação. Clique em *Add Reference* para que a operação seja completada. Observe que na *Solution Explorer* aparece um novo item: *Web References* como um arquivo cujo nome é o mesmo digitado na janela anterior. Se desejar visualizar o código gerado automaticamente, pelo *Windows Explorer*, abra a pasta da aplicação *Windows* e abra a pasta *Service1* que irá conter os arquivos necessários. O código fonte gerado automaticamente pelo *Visual Studio* está no arquivo *References.cs*.

Com este passo concluído, vamos escrever o código necessário para consumir nosso *Web Service*. Dê um duplo clique no botão e adicione o código da Listagem 3.

Listagem 3. Código para executar o Web Service dentro do Windows Form

```
private void btnExecutar_Click(object sender, EventArgs e)
{
    // Define variáveis para a operação.
    decimal Numero1 = 0;
    decimal Numero2 = 0;
    decimal Resultado = 0;
    string op = "";
    // Variável para tratamento dos erros
    string erro = "";

    // Faz a conversão dos números digitados
    try
    {
        Numero1 = decimal.Parse(txtNumero1.Text);
        Numero2 = decimal.Parse(txtNumero2.Text);
    }
    catch
    {
        // Se houver erro, informa ao usuário
        MessageBox.Show("Números digitados inválidos.",
            "Atenção", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Instancia o Web Service
    service1.Service1 servico = new AppWin.service1.Service1();
    // Configura o endereço (URL) do Web Service
    servico.Url = "http://localhost/webservice/service1.asmx";

    // Verifica qual a operação foi selecionada
    switch (cmbOperacao.SelectedIndex)
    {
        case 0: op = "+"; break;
        case 1: op = "-"; break;
        case 2: op = "x"; break;
        case 3: op = "/"; break;
    }

    // Chama o Web Service
    try
    {
        Resultado = servico.OpArit(Numero1, Numero2, op, out erro);

        // O Web Service foi executado, verifica se houve erros
        if (erro == "")
        {
            // Exibe o resultado
            txtResultado.Text = Resultado.ToString();
        }
        else
        {
            MessageBox.Show("Erro durante a operação:\n" + erro,
                "Atenção", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    catch (Exception ex)
    {
        // Trata erros ocorridos
        MessageBox.Show("Não foi possível consultar o Web
            Service\n" + ex.Message, "Atenção", MessageBoxButtons.OK,
            MessageBoxIcon.Warning);
    }
}
}
```

Listagem 4. Código HTML correspondente ao formulário

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Aplicação ASP.NET</title>
</head>
<body>
<form id="form1" runat="server">
<div style="width:100%; background-color: Navy;
font-family: arial; font-weight:bold; font-size:x-large;
color:White;">
<br />
Consumindo Web Services com ASP.NET
<br /><br />
</div>
<br />
<div>
<asp:Label ID="Label1" runat="server" Text="Número 1" />
&nbsp;
<asp:TextBox ID="txtNumero1" runat="server" Columns=
"20" /><br />
<asp:Label ID="Label2" runat="server" Text=
"Número 2" />
&nbsp;
<asp:TextBox ID="txtNumero2" runat="server"
Columns="20" /><br />
<asp:Label ID="Label3" runat="server" Text="Operação" />
&nbsp;
<asp:DropDownList ID="cmbOperacao" runat="server">
<asp:ListItem>Soma</asp:ListItem>
<asp:ListItem>Subtração</asp:ListItem>
<asp:ListItem>Multiplicação</asp:ListItem>
<asp:ListItem>Divisão</asp:ListItem>
</asp:DropDownList><br />
<asp:Label ID="Label4" runat="server" Text="
Resultado" />
&nbsp;
<asp:TextBox ID="txtResultado" runat="server"
Columns="20" ReadOnly="true"/><br
/><br />
<asp:Button ID="btnOperacao" runat="server" Text=
"Executar"
OnClick="btnOperacao_Click" />
<br />
<br />
<asp:Panel ID="Panel1"
runat="server"
Width="100%"
BorderColor="gray"
BackColor="Khaki"
HorizontalAlign="Center"
Visible="false">
<asp:Label ID="lblErro" Font-Size="X-Small" Font-
Names="Verdana"
runat="server" Text="" />
</asp:Panel>
</div>
</form>
</body>
</html>
```



Figura 24. Aplicação Windows consumindo o Web Service

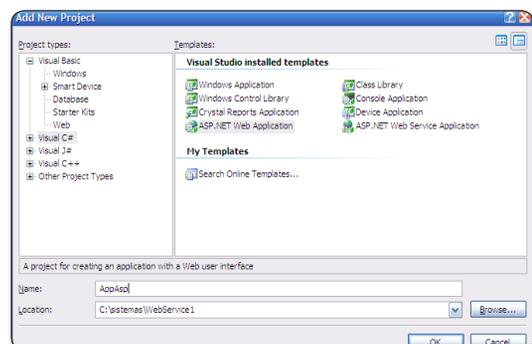


Figura 25. Criando a aplicação ASP.NET no Visual Studio

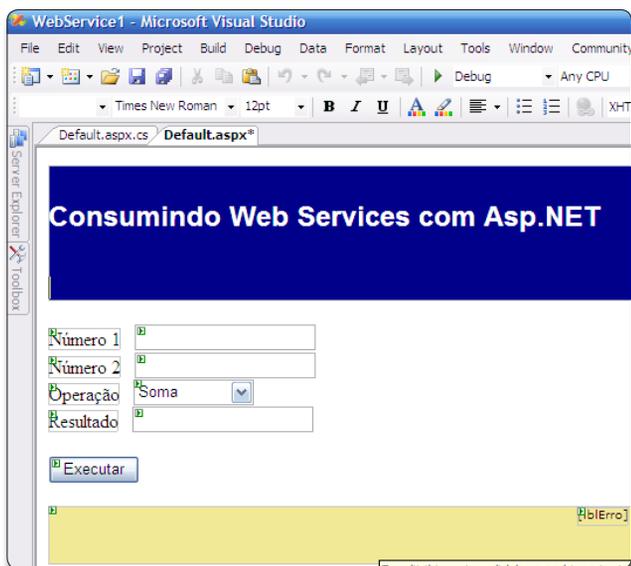


Figura 26. Design da página da aplicação Web

Solution Explorer aparece um novo item: *Web References*, como um arquivo cujo nome é o mesmo digitado na janela anterior. Note que este processo é semelhante ao usado para adicionar uma *Web Reference* num projeto *Windows*. Para mais detalhes, reveja a **Figura 23**.

Para escrever o código em C# para invocar o *Web Service* dê um duplo clique no botão *Executar* e escreva o código como na **Listagem 5**.

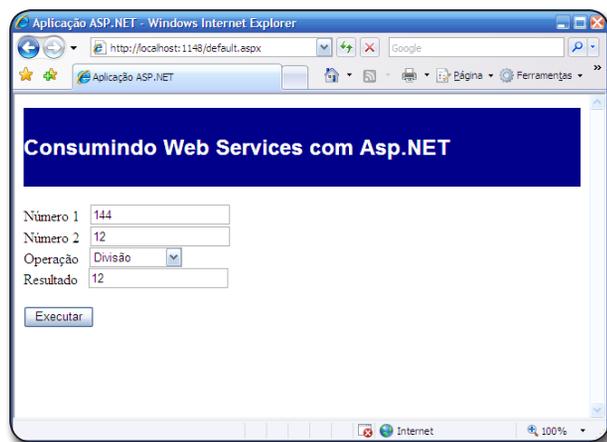


Figura 27. Aplicação Web sendo executada

Conclusão

O exemplo apresentado serve como base para *Web Services* mais completos. Pode-se criar por exemplo um *Web Service* para executar tarefas de consulta a banco de dados ou processamento diversos. A implementação tanto para aplicações *Windows* como *ASP* seguirá o modelo visto neste artigo podendo variar em alguns detalhes. ●

Listagem 5. Código para executar o Web Service

```
protected void btnOperacao_Click(object sender, EventArgs e)
{
    lblErro.Text = "";
    this.Panel1.Visible = false;
    // Define variáveis para a operação.
    decimal Numero1 = 0;
    decimal Numero2 = 0;
    decimal Resultado = 0;
    string op = "";
    // Variável para tratamento dos erros
    string erro = "";

    // Faz a conversão dos números digitados
    try
    {
        Numero1 = decimal.Parse(txtNumero1.Text);
        Numero2 = decimal.Parse(txtNumero2.Text);
    }
    catch
    {
        // Se houver erro, informa ao usuário
        lblErro.Text = "Números inválidos.";
        this.Panel1.Visible = true;
        return;
    }

    // Instancia o Web Service
    service1.Service1 servico = new AppAsp.service1.Service1();
    // Configura o endereço (URL) do Web Service
    servico.Url = "http://localhost/webservice/service1.asmx";

    // Verifica qual a operação foi selecionada
    switch (cmbOperacao.SelectedIndex)
    {
        case 0: op = "+"; break;
        case 1: op = "-"; break;
        case 2: op = "x"; break;
        case 3: op = "/"; break;
    }

    // Chama o Web Service
    try
    {
        Resultado = servico.OpArit(Numero1, Numero2, op, out erro);

        // O Web Service foi executado, verifica se houve erros
        if (erro == "")
        {
            // Exibe o resultado
            txtResultado.Text = Resultado.ToString();
        }
        else
        {
            lblErro.Text = "Erro durante a operação: " + erro;
            this.Panel1.Visible = true;
        }
    }
    catch (Exception ex)
    {
        // Trata erros ocorridos
        lblErro.Text = "Não foi possível executar." + erro;
        this.Panel1.Visible = true;
    }
}
```

Dê seu feedback sobre esta edição!

A .NET Magazine tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/netmagazine/feedback

