Ambientes de Desenvolvimento Avançados

http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm

Aula 17

Engenharia Informática

2004/2005

José António Tavares
jrt@isep.ipp.pt

.NET Web Services:

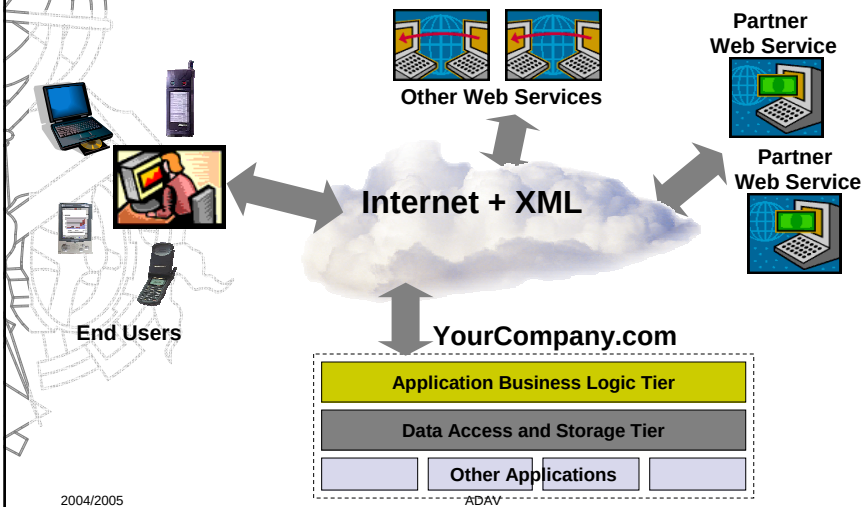
Construção de um Serviço WEB Simples

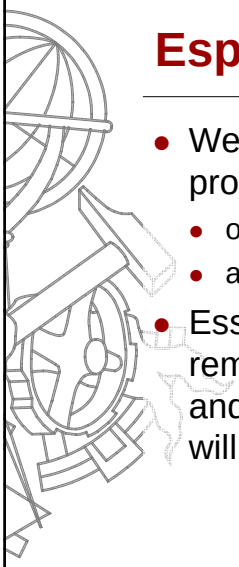
Resumo

- The goal for this presentation is to build our very first Web service!
- We will be playing the role of the Web service producer for this talk.
- We will see how **easy** creating Web services are with Visual Studio .NET.

Web Services Overview

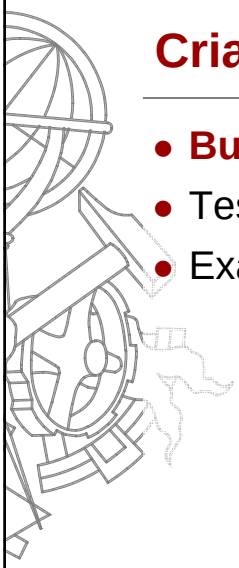
Application Model





Especificação do Serviço WEB

- We will build a very simple Web Service that provides two methods:
  - one that **adds** to integers,
  - and one that **subtracts** two integers.
- Essentially our service will sit there, waiting for remote computers to ask, “What is the sum of *x* and *y*,” where *x* and *y* are integers. Our service will then response, “The answer is *z*.”

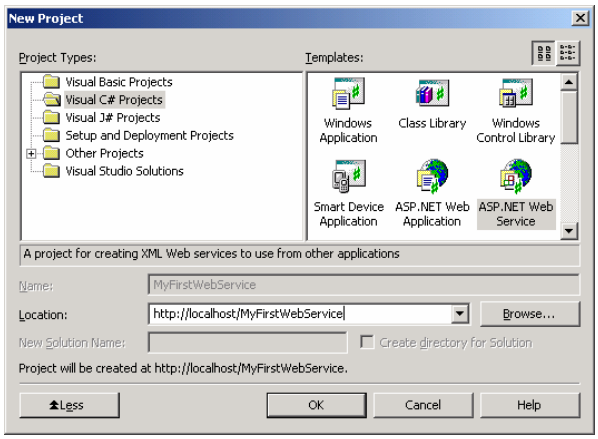


Criação de um Serviço WEB

- **Building the Web Service (VS.NET)**
- Testing the Web Service
- Examining the WSDL Document

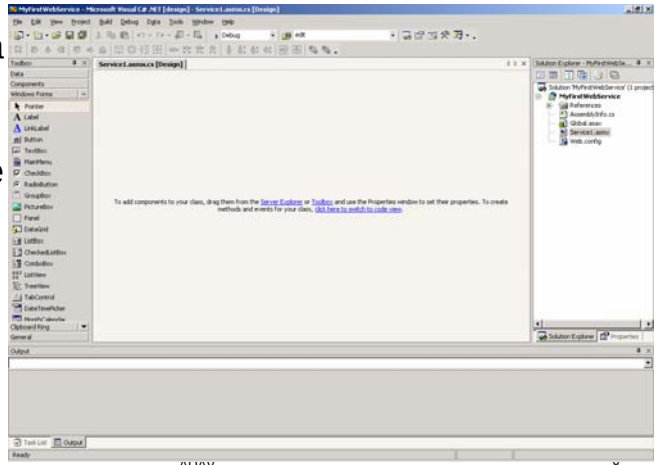
Creating a Web Service with Visual Studio .NET

To create a Web Service in Visual Studio .NET, choose to create a new project type of ASP.NET Web Service.



Creating a Web Service with Visual Studio .NET

This will create a new ASP.NET Web Application project with a file called **`Service1.asmx`**



Creating a Web Service with Visual Studio .NET

- Things to note/realize:
  - Web services are located on a Web server and are accessible via a URL.
  - .NET Web services use the file extension **`.asmx`**.
  - .NET Web services are actually compiled classes. Particular methods of this class are the methods the Web service exposes.

Creating a Web Service with Visual Studio .NET

- To see the Web service class's code, simply right-click in the Design window and choose, View Code.
- This will display the file **`Service1.asmx.cs`**, whose code can be seen on the next slide.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace MyFirstWebService
{
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            InitializeComponent();
        }
        …

//      [WebMethod]
//      public string HelloWorld()
//      {
//          return "Hello World";
//      }
    }
}
```


Web Service Class

- Things to note:
  - The class is given the same name as the Web service file (Service1).
  - The class is derived (inherited) from the **`System.Web.Services.WebService`** class.
  - The class contains an example Web service method in comments, **`HelloWorld()`**.

Adding Methods to the Web Service Class

- Recall that a Web service exposes a set of methods. These are methods that are callable by clients consuming the Web service.
- Adding such methods to the Web service is incredibly easy – first, just create a method like you normally would.

Adding Methods to the Web Service Class

- For example, to create our **Add()** method that will add two Integers, we'd use the following code:

```
public int Add(int x, int y)
{      // C#
  return x + y;
}

' VB.NET
Public Function Add(x as Integer, y as Integer) as Integer
  Return x + y
End Function
```

Adding Methods to the Web Service Class

- Next, add the `WebMethod()` attribute immediately prior to the method declaration.

```
[WebMethod()]
public int Add(int x, int y) { ... }

<WebMethod()>_
Public Function Add(x as Integer, y as Integer) as Integer
  ...
End Function
```

Note the line continuation character after <WebMethod()> in the VB.NET example

Adding Methods to the Web Service Class

- That's all there is to it!!!
- For each method in our class that we want to be available for the Web service, we simply add that **`WebMethod()`** attribute.
- If you are following along, go ahead and add the **`Subtract()`** method as well.

The Two Methods in VB.NET

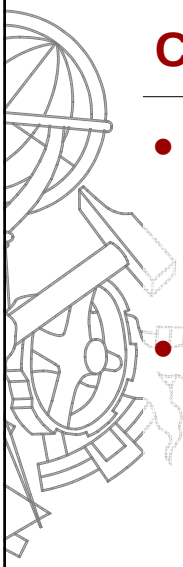
```
<WebMethod()> _
Public Function Add(x as Integer, y as Integer) _
as Integer
    Return x + y
End Function

<WebMethod()> _
Public Function Subtract(x as Integer, y as Integer) _
as Integer
    Return x - y
End Function
```


The Two Methods, in C#

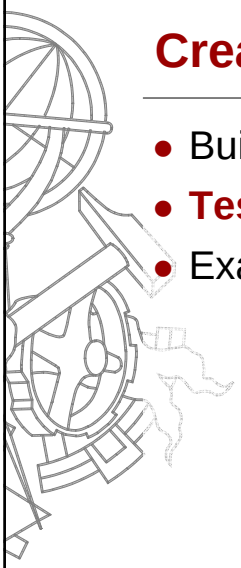
```
[WebMethod()]
public int Add(int x, int y)
{
    return x + y;
}

[WebMethod()]
public int Subtract(int x, int y)
{
    return x - y;
}
```



Compiling the Web Service

- Once you have added the **`Add()`** and **`Subtract()`** methods to the Web service, you need to compile the Web service class.
- To accomplish this, go to Build and choose Build Solution.



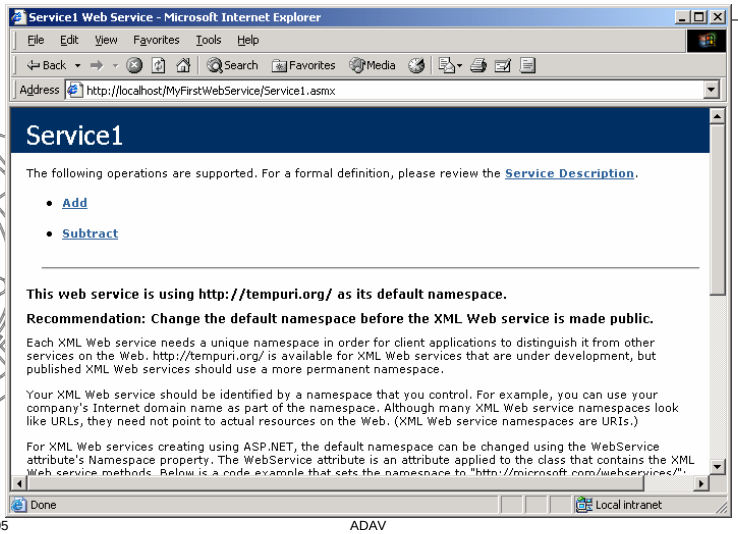
Creating a Web Service

- Building the Web Service (VS.NET)
- **Testing the Web Service**
- Examining the WSDL Document

Testing the Web Service

- At this point, the Web service is complete and ready for consumption.
- If you visit the Web service directly through a Web browser, ASP.NET provides human-friendly interface to the Web service, known as the service description page. This page can be used as a means to test the Web service.
- Launch a browser and visit the Web service:
  `http://localhost/MyFirstWebService/Service1.asmx`

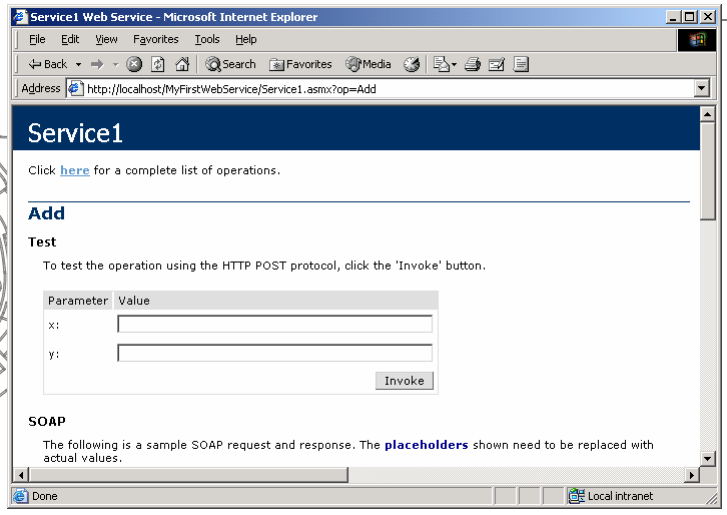
Testing the Web Service



Testing the Web Service

- Note that the service description page provides links for the Web service's two methods.
- Clicking on either of these method names takes us to a description for the particular method.
- This description page lists the means by which the Web service can be invoked, and provides a testing interface.

Testing the Web Service

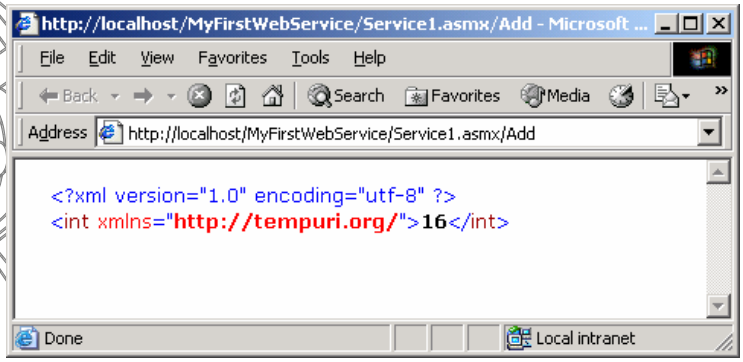


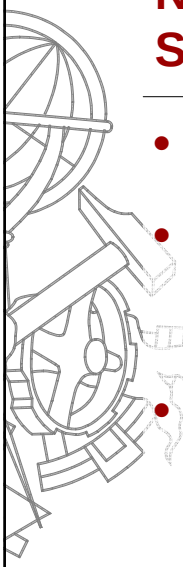
Testing the Web Service

- Enter two integer values into the X and Y textboxes and click Invoke.
- This will invoke the Web service, passing in the two parameters you specify.
- The Web service's response will open in a new browser window.

Testing the Web Service

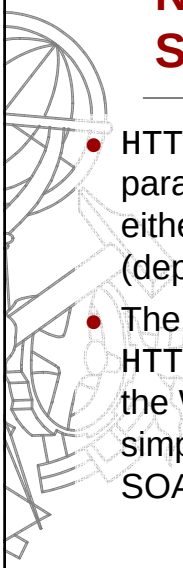
- The Web service's response from calling it with the parameters 4 and 12:





Notes on Testing the Web Service

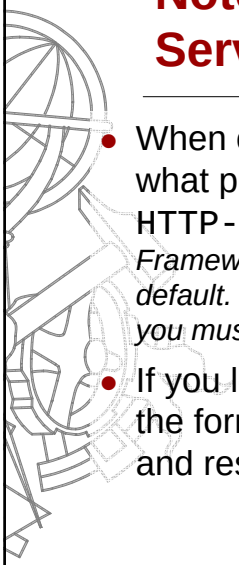
- Web services use **SOAP** as the message format protocol.
- That is, messages being sent to and from a Web service are encoded into a SOAP message, with a SOAP envelope and SOAP body.
- However, there are other, simpler message format protocols in addition to SOAP: HTTP-GET and HTTP-POST.



Notes on Testing the Web Service

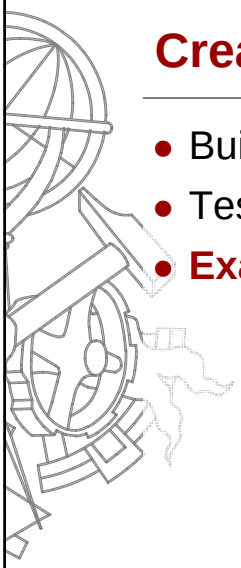
- `HTTP-GET` and `HTTP-POST` send the input parameters and Web service method output through either the `QueryString` or `HTTP POST` headers (depending on whether `GET` or `POST` is being used).
- The testing interface in the Web page uses `HTTP-POST` – this is why the output received from the Web service we tested a few slides ago is a simple one line response, and **not** a complete SOAP envelope.

Notes on Testing the Web Service



- When creating a Web service, you can specify what protocols it accepts (`HTTP-GET`, `HTTP-POST`, and/or SOAP). *(Note that with the .NET Framework 1.1, only the SOAP protocol is supported by default. To provide `HTTP-GET` or `HTTP-POST` support you must explicitly specify this)*
- If you look at the Web service description page, the format of the SOAP and `HTTP-POST` request and response messages are spelled out.

Creating a Web Service



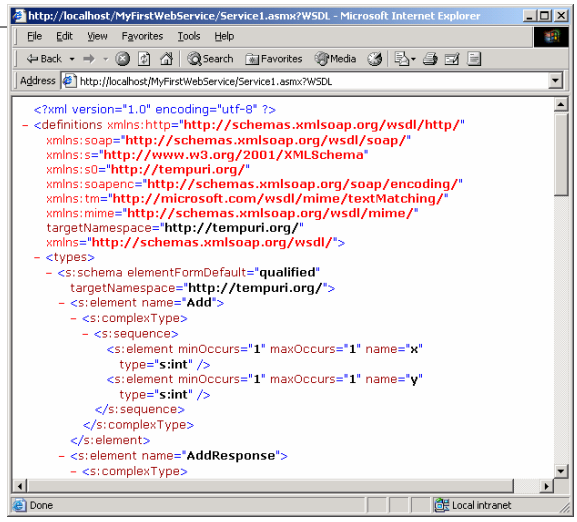
- Building the Web Service (VS.NET)
- Testing the Web Service
- **Examining the WSDL Document**

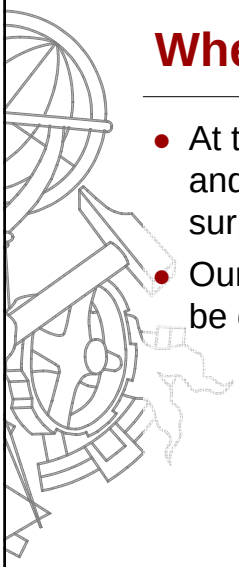
The Web Service's WSDL Document

- All Web services contain a **WSDL** file (**W**eb **S**ervice **D**escription **L**anguage) that very precisely spells out the Web service's methods, their input and output parameters, how the Web service can be invoked (HTTP-GET/HTTP-POST/SOAP), and other such information.
- The Web service description page contains a link to the Web service's WSDL file (go to the first page and click the "Service Description" link)

The Web Service's WSDL Document

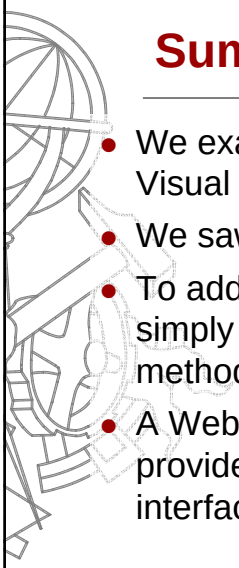
- The WSDL document is an XML-formatted document.
- We'll discuss WSDL in greater detail later on!





Where We're At Now!

- At this juncture, we've just completed creating and testing our very first Web service. Wasn't it surprisingly easy?
- Our Web service is now available and ready to be consumed!



Summary

- We examined how to build a Web service with Visual Studio .NET.
- We saw that a Web service is actually a class.
- To add invokable methods to a Web service, we simply add the **`WebMethod()`** attribute before the method in the Web service class.
- A Web service description page is available, which provides WSDL information and a testable interface to the Web service's method(s).

.NET Web Services:
Consumo de um Serviço WEB Simples

Resumo

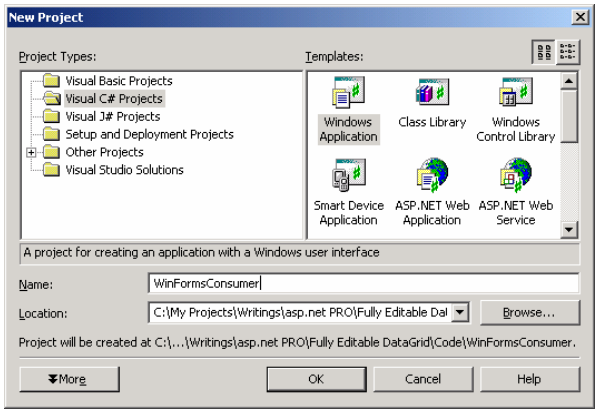
- The goal for this presentation is to consume the Web service we built in the previous presentation.
- We will be playing the role of the Web service consumer for this talk.
- We will see how **easy** consuming Web services are with Visual Studio .NET.

Mediums for Consumption

- Web services can be consumed through both desktop applications and Web applications.
- The process of consuming the Web service is the same for either approach.
- We'll examine creating two consuming applications in this talk:
  1. A C# WinForms desktop application
  2. A VB.NET ASP.NET Web application

Consumption through WinForms

To start, create a new Visual Studio .NET Windows application (*feel free to use VB.NET if you like*).

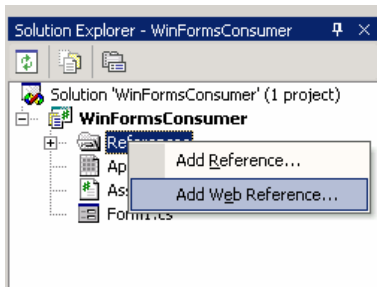


Designing the User Interface

- Recall that our Web service has two methods:
  - `Add(`*x*`, `*y*`)`, and
  - `Subtract(`*x*`, `*y*`)`
- For the UI, add two textboxes (for the values of *x* and *y*) and two buttons, one labeled “Add” and one labeled “Subtract”
- Finally, create a label for the answer.

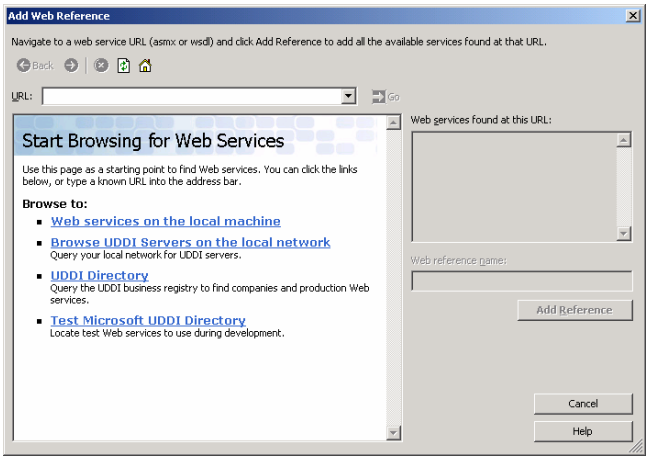
Consuming the Web Service

- In order to consume a Web service, we must add a Web Reference to the Web Service.
- Right click on the References folder in the Solution Explorer and choose “Add Web Reference.”



Consuming the Web Service

- Adding a Web Reference will display the Add Web Reference dialog box:



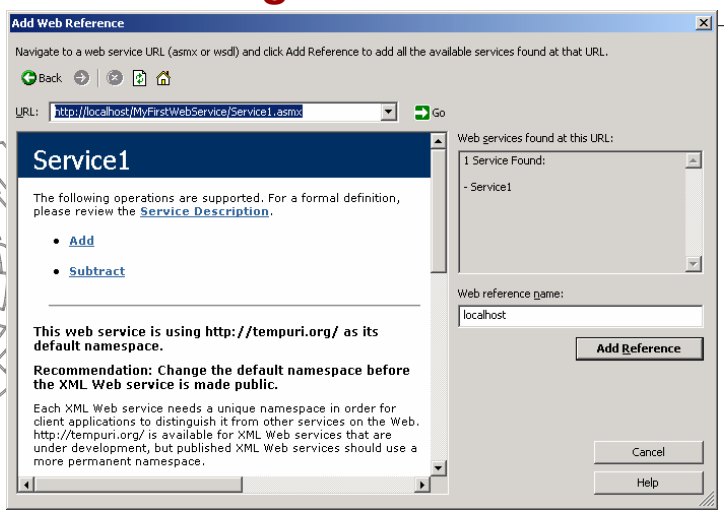
Consuming the Web Service

- In the Add Web Reference dialog box, enter the URL of the Web Service we created from the previous talk:

  `http://localhost/MyFirstWebService/Service1.asmx`

- This will display the Web service's description in the left side window pane and allow you to specify the Web reference name.

Consuming the Web Service



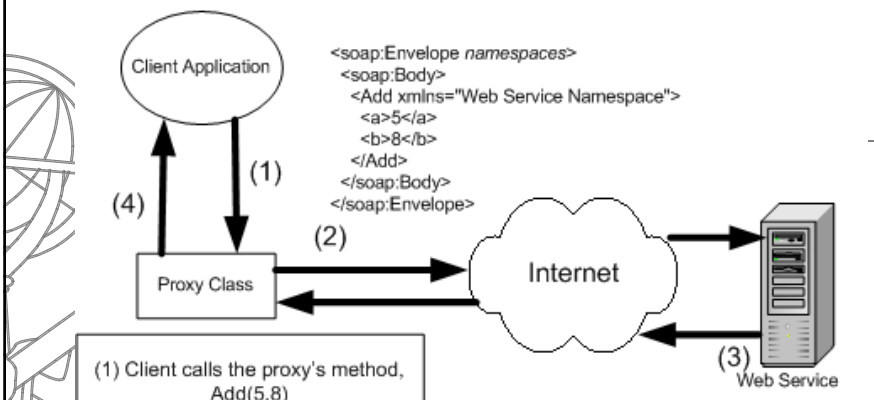
Consuming the Web Service

- Change the Web reference name to `Calculator`, and then click the Add Reference button.
- This will create a new namespace in your project called **`Calculator`**. This namespace will contain a class named **`Service1`** that has the methods from the Web service – **`Add(int, int)`** and **`Subtract(int, int)`**.

Consuming the Web Service

- Adding a Web reference actually creates what is called a ***proxy class*** on your computer.
- This proxy class has all of the methods of the remote Web service. These methods, when called, invoke the Web service.
- Hence, you can call the Web service as if it were a local component.

We'll discuss proxy classes in much greater detail in future presentations!



Consuming the Web Service

- In the Add button's Click event handler, simply create a new instance of the proxy class and call the Add method:

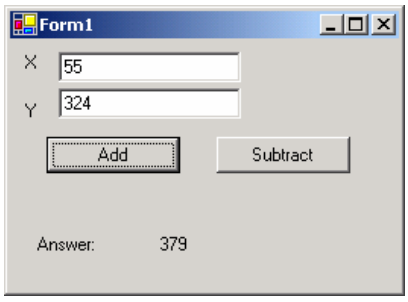
```
private void btnAdd_Click(object sender, System.EventArgs e)
{
        int x = Convert.ToInt32(txtX.Text);
        int y = Convert.ToInt32(txtY.Text);

        Calculator.Service1 calc = new Calculator.Service1();
        int result = calc.Add(x,y);

        this.lblAnswer.Text = result.ToString();
}
```


Consuming the Web Service

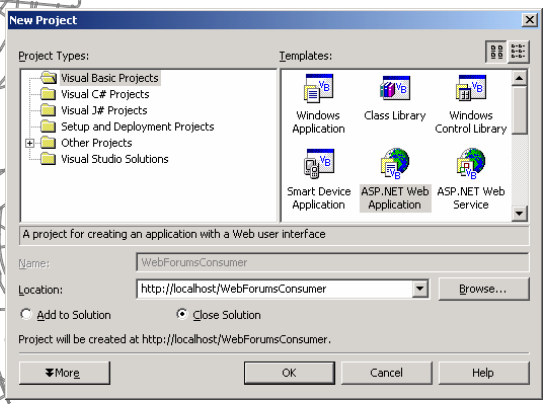
- Similar code goes in the Subtract button's Click event handler.
- Testing the application out, we see we get the desired results:



Consumption through an ASP.NET Web Application

- Consuming a Web Service through an ASP.NET Web application is strikingly similar.
- Essentially, we just add a Web Reference, in the same manner, and then add the needed code in the proper location of the ASP.NET Web page's code-behind class.

Consumption through an ASP.NET Web Application



Create a new Visual Studio .NET ASP.NET Web application project named **`WebFormsConsumer`**

Consumption through an ASP.NET Web Application

- Create the WebForm UI – again, use two textboxes for the X & Y inputs, two buttons for the Add/Subtract options, and a Label to display the answer.
- Add a Web Reference as before, again setting the Web reference name to **`Calculator`**.

Consumption through an ASP.NET Web Application

- Add the appropriate code for the **`Add`** and **`Subtract`** Click event handlers.

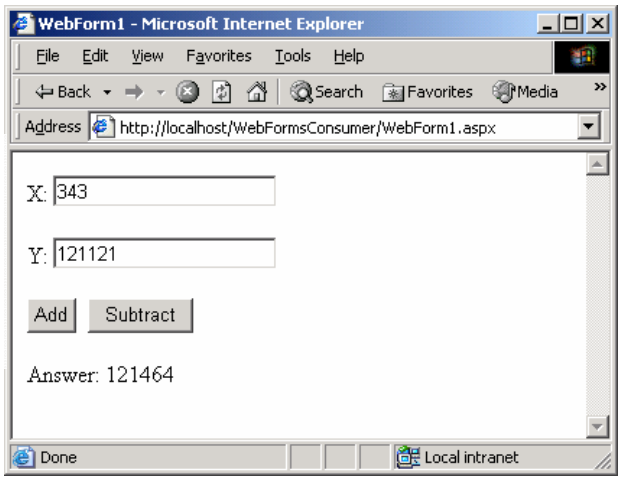
```
Private Sub btnAdd_Click(ByVal sender As System.Object, _
                         ByVal e As System.EventArgs) _
                                 Handles btnAdd.Click
    Dim x As Integer = Convert.ToInt32(txtX.Text)
    Dim y As Integer = Convert.ToInt32(txtY.Text)

    Dim calc As New Calculator.Service1

    Dim result As Integer = calc.Add(x, y)
    Me.lblAnswer.Text = result.ToString()
End Sub
```

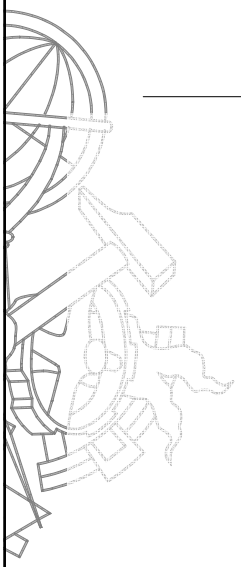
Consumption through an ASP.NET Web Application

This Web page demonstrates the Web service in action!



Summary

- In this talk we saw how to use Visual Studio .NET to consume a Web service.
- This process was simple, all we had to do was create a Web Reference and then we could call the Web service as if it was a local component.
- We'll examine the details of Web service consumption in a future talk.



Questões

?