



## Desarrollo de Aplicaciones Distribuidas

### AUTORES:

Alejandro Calderón Mateos

Javier García Blas

David Expósito Singh

Laura Prada Camacho

Departamento de Informática  
Universidad Carlos III de Madrid  
Julio de 2012

**DESARROLLO DE APLICACIONES DISTRIBUIDAS**

**CON .NET:**

**SERVICIOS WEB EN .NET**

# Contenidos

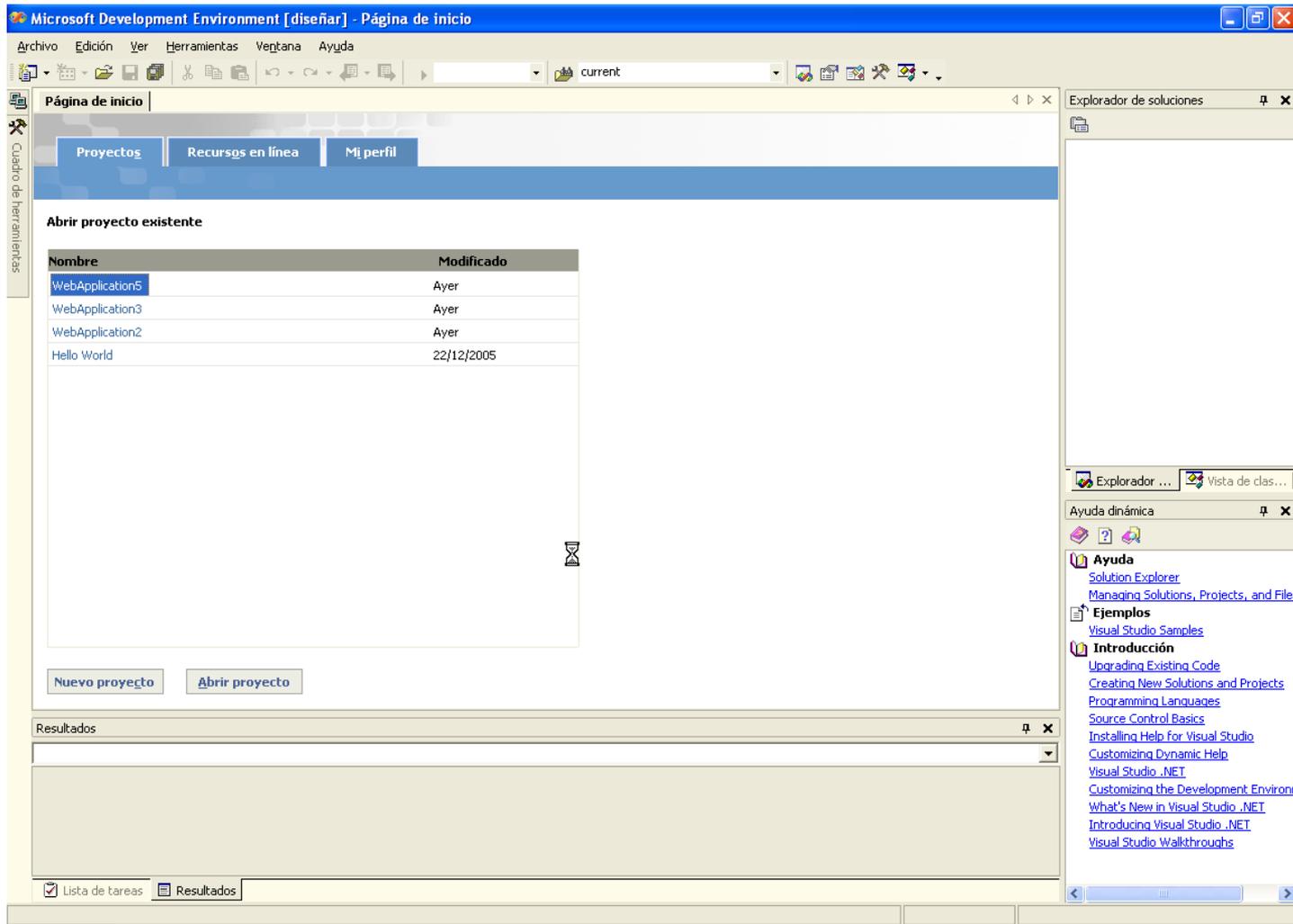
1. Introducción Visual Studio .NET
2. Ejemplo con .NET SDK
3. Ejemplo con Visual Studio 2008



# Contenidos

1. Introducción a Visual Studio .NET
2. Ejemplo con .NET SDK
3. Ejemplo con Visual Studio 2008

# Introducción a Visual Studio .NET



## Unidades en Visual Studio .NET

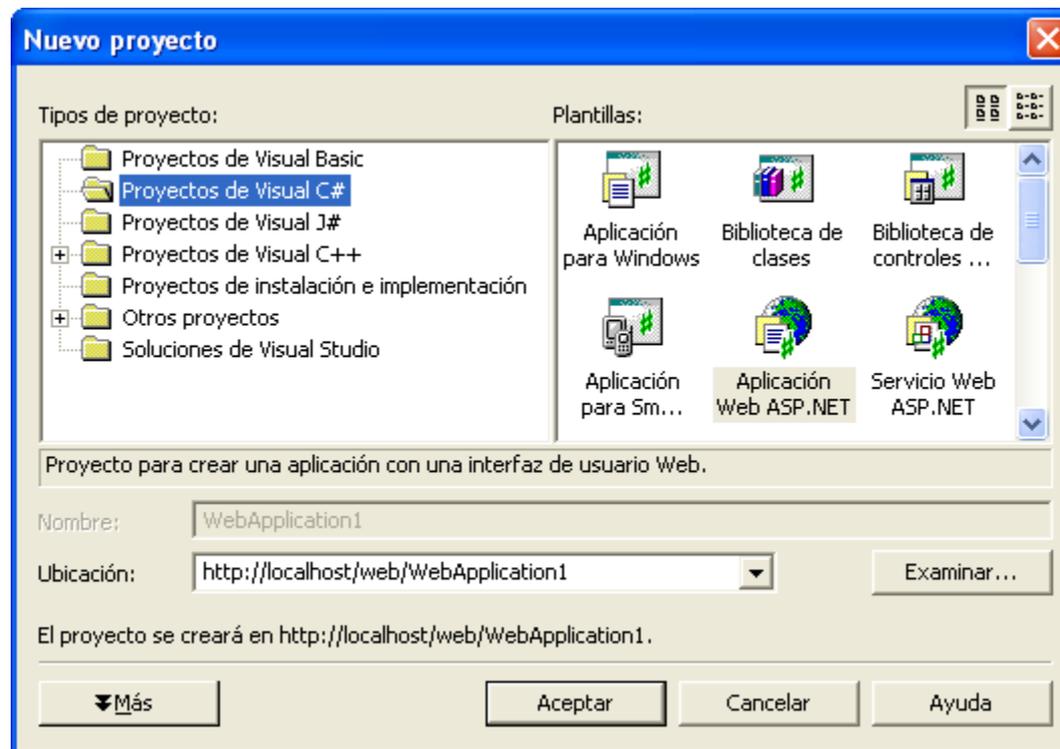
### □ Soluciones

- ▣ Una solución es el punto de inicio para la creación de aplicaciones con Visual Studio
- ▣ Es el contenedor que permite almacenar todas las partes individuales que formarán la aplicación
- ▣ Una solución contiene uno o más proyectos

### □ Proyectos

- ▣ Un proyecto se puede crear utilizando cualquier lenguaje de Visual Studio .NET.
- ▣ Contiene una serie de ficheros
- ▣ El IDE organiza soluciones, proyectos y ficheros de un trabajo de forma jerárquica

## Distintos tipos de proyectos



# Entorno integrado de desarrollo (IDE)

Cuadro de herramientas

Editor/Navegador

Examinador de objetos

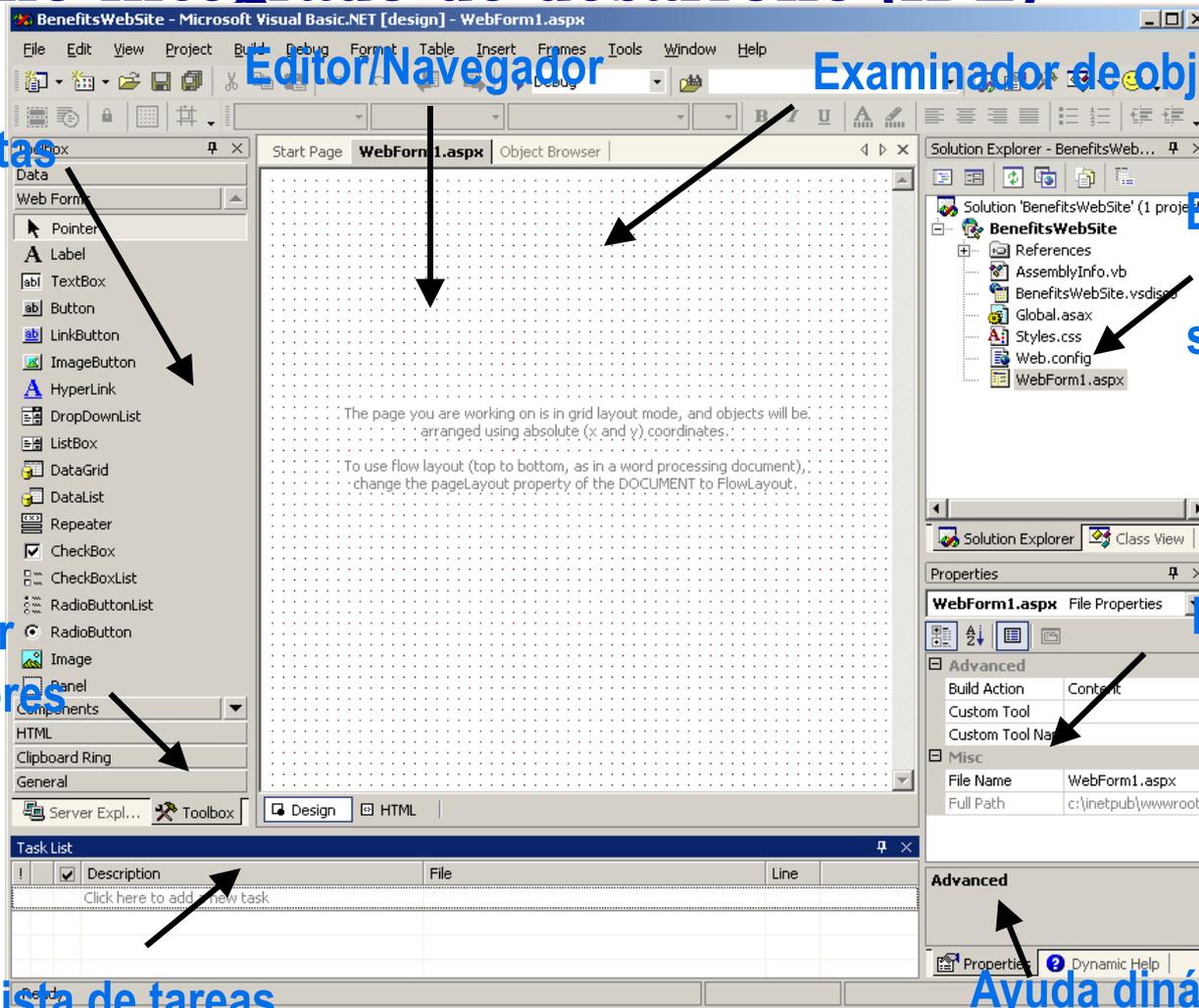
Explorador de soluciones

Explorador de servidores

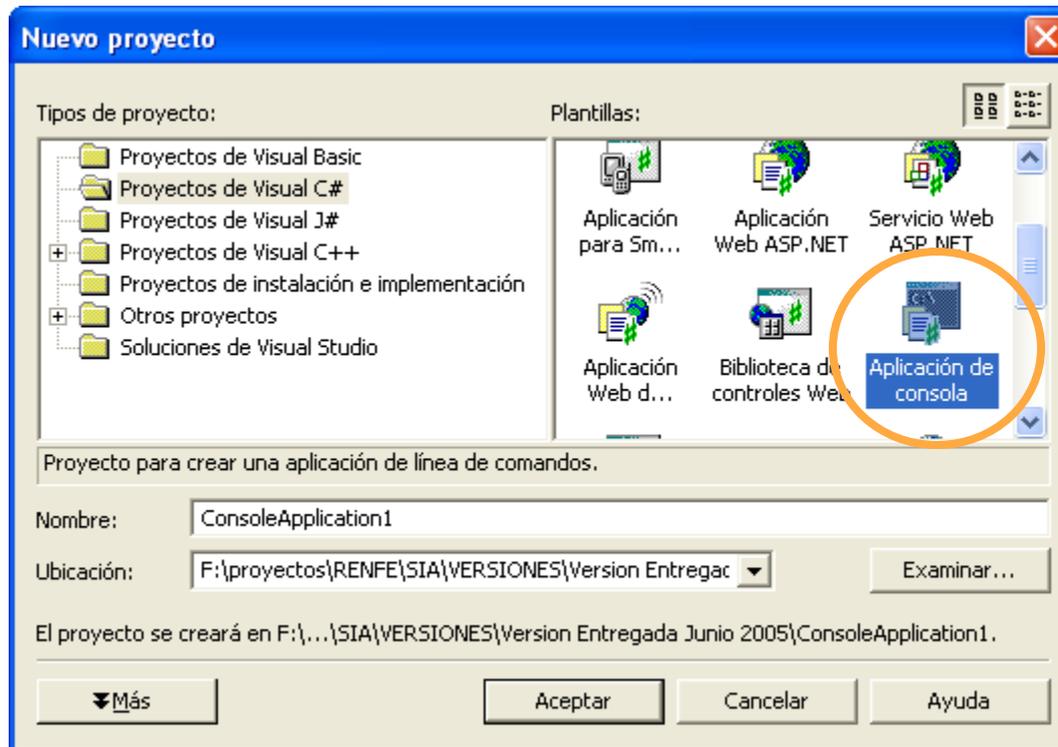
Propiedades

Lista de tareas

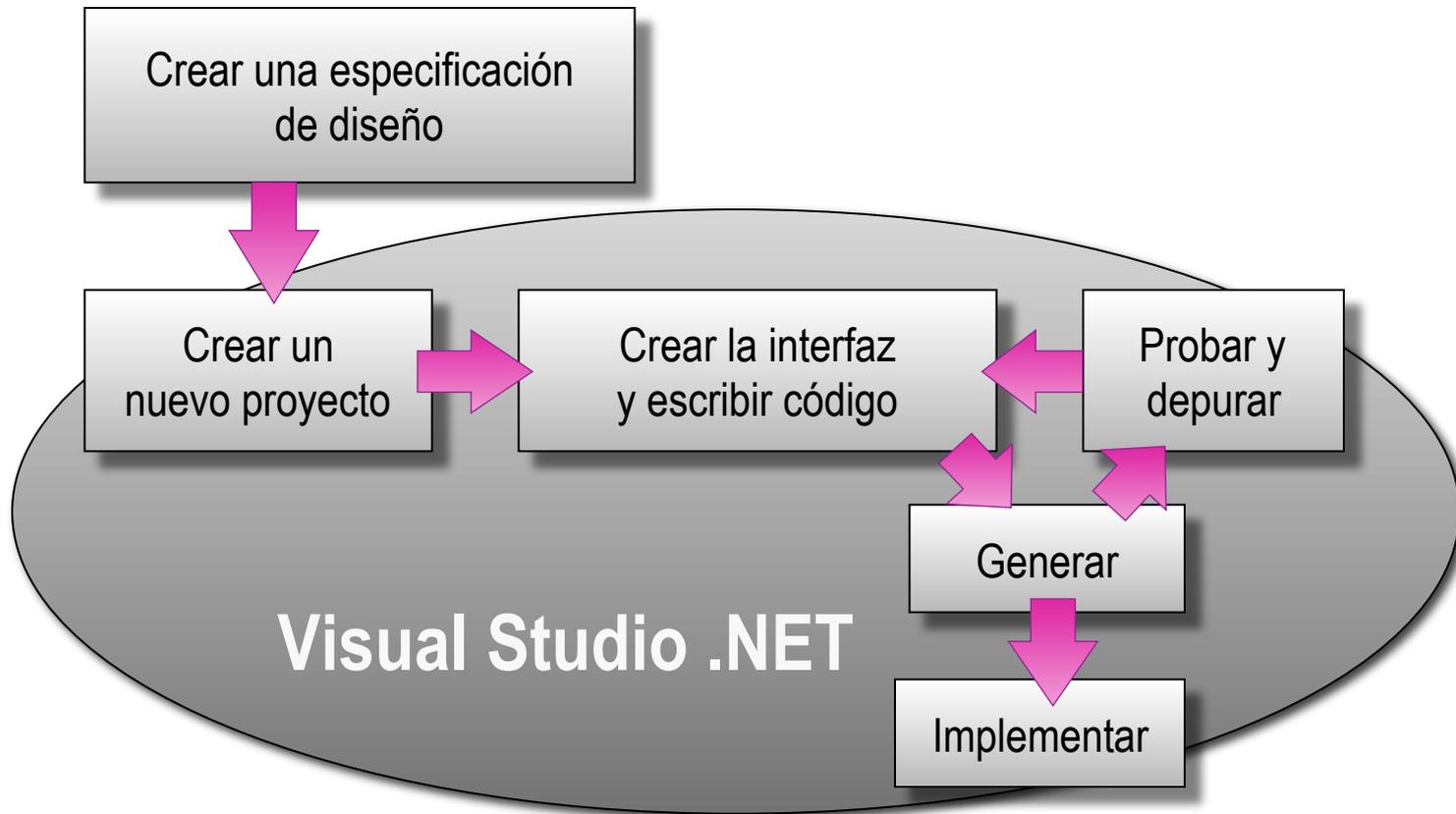
Ayuda dinámica



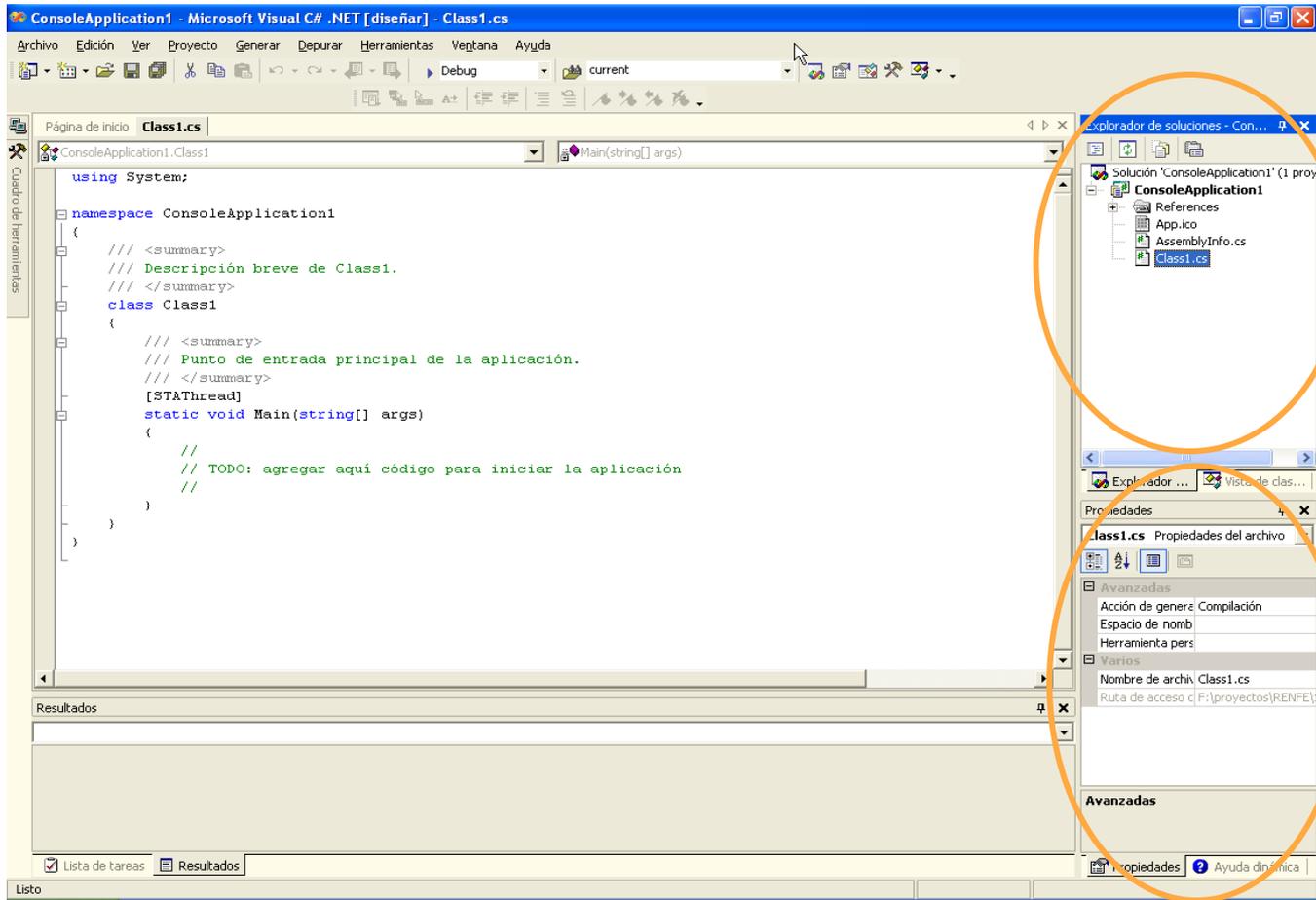
## Ejemplo: Aplicación de consola



## El proceso de desarrollo



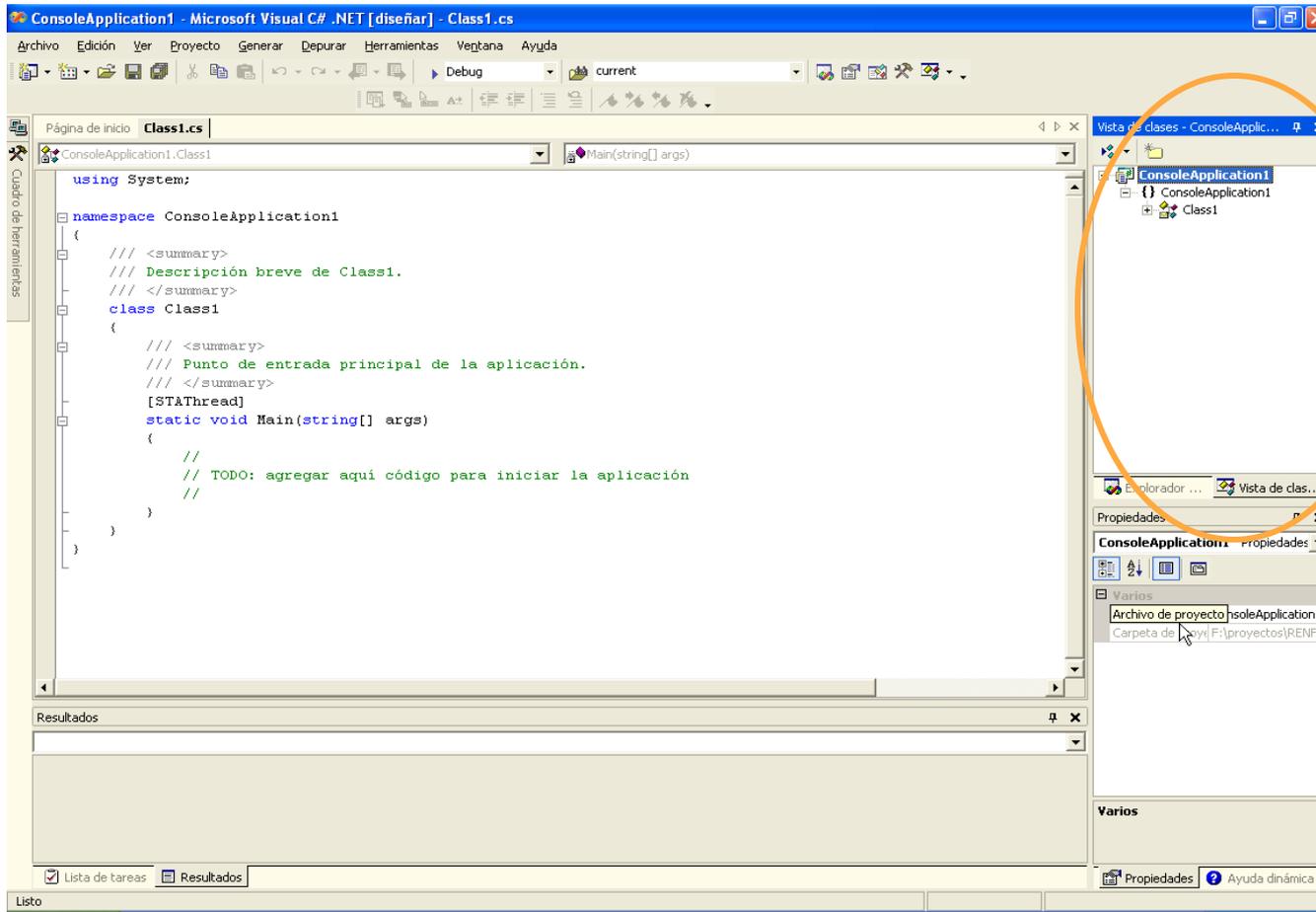
# Exploradores



Explorador de  
Soluciones

Explorador de  
Propiedades

# Vista de clases



Vista de  
clases

# Creación de un método nuevo

ConsoleApplication1 - Microsoft Visual C# .NET [diseñar] - Class1.cs

Archivo Edición Ver Proyecto Generar Depurar Herramientas Ventana Ayuda

Página de inicio Class1.cs

```
using System;  
  
namespace ConsoleApplication1  
{  
    /// <summary>  
    /// Descripción breve de la clase.  
    /// </summary>  
    class Class1  
    {  
        /// <summary>  
        /// Punto de entrada principal para el programa.  
        /// </summary>  
        [STAThread]  
        static void Main()  
        {  
            // TODO: Agregar el código de inicio aquí.  
        }  
    }  
}
```

**Asistente para métodos de C# - ConsoleApplication1**

Asistente para agregar métodos de C#  
Este asistente agrega un método a la clase de C#.

Acceso al método: public Tipo de valor devuelto: void Nombre de método: HolaMundo

Modificador: No Tipo de parámetro: int Nombre de parámetro: Lista de parámetros:

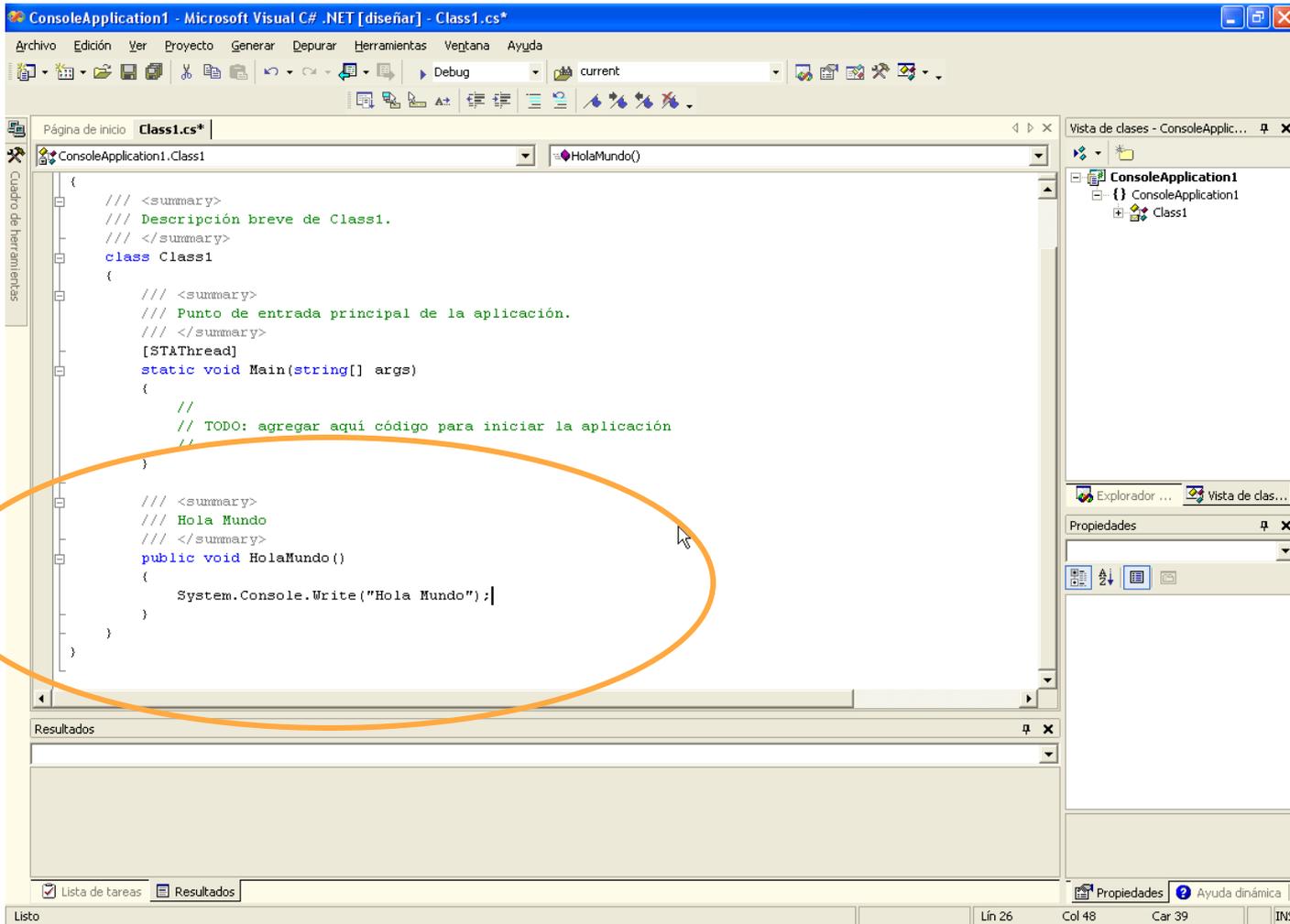
Modificadores del método:  
 Static  Abstract  Virtual  Extern  Override  New

Comentario (notación // no requerida):  
Hola Mundo

Firma del método:  
public void HolaMundo(void){ }

Finalizar Cancelar Ayuda

# Inserción de código



# Compilación

ConsoleApplication1 - Microsoft Visual C# .NET [diseñar] - Class1.cs\*

Archivo Edición Ver Proyecto Generar Depurar Herramientas Ventana Ayuda

Generar solución Ctrl+Mayús+B  
Volver a generar solución  
Generar ConsoleApplication1  
Volver a generar ConsoleApplication1  
Generación por lotes...  
Administrador de configuración...

```
using System;

/// <summary>
/// Descripción breve de Class1.
/// </summary>
class Class1
{
    /// <summary>
    /// Punto de entrada principal de la aplicación.
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
        //
        // TODO: agregar aquí código para iniciar la aplicación
        //
        HolaMundo();
    }

    /// <summary>
    /// Hola Mundo
    /// </summary>
    public void HolaMundo()
    {
        System.Console.WriteLine("Hola Mundo");
    }
}
```

Vista de clases - ConsoleApplic...

ConsoleApplication1  
- ConsoleApplication1  
+ Class1

Propiedades

Resultados

Listo Col 25 Car 16

# Resultado de la compilación

The screenshot displays the Visual Studio IDE with the following components:

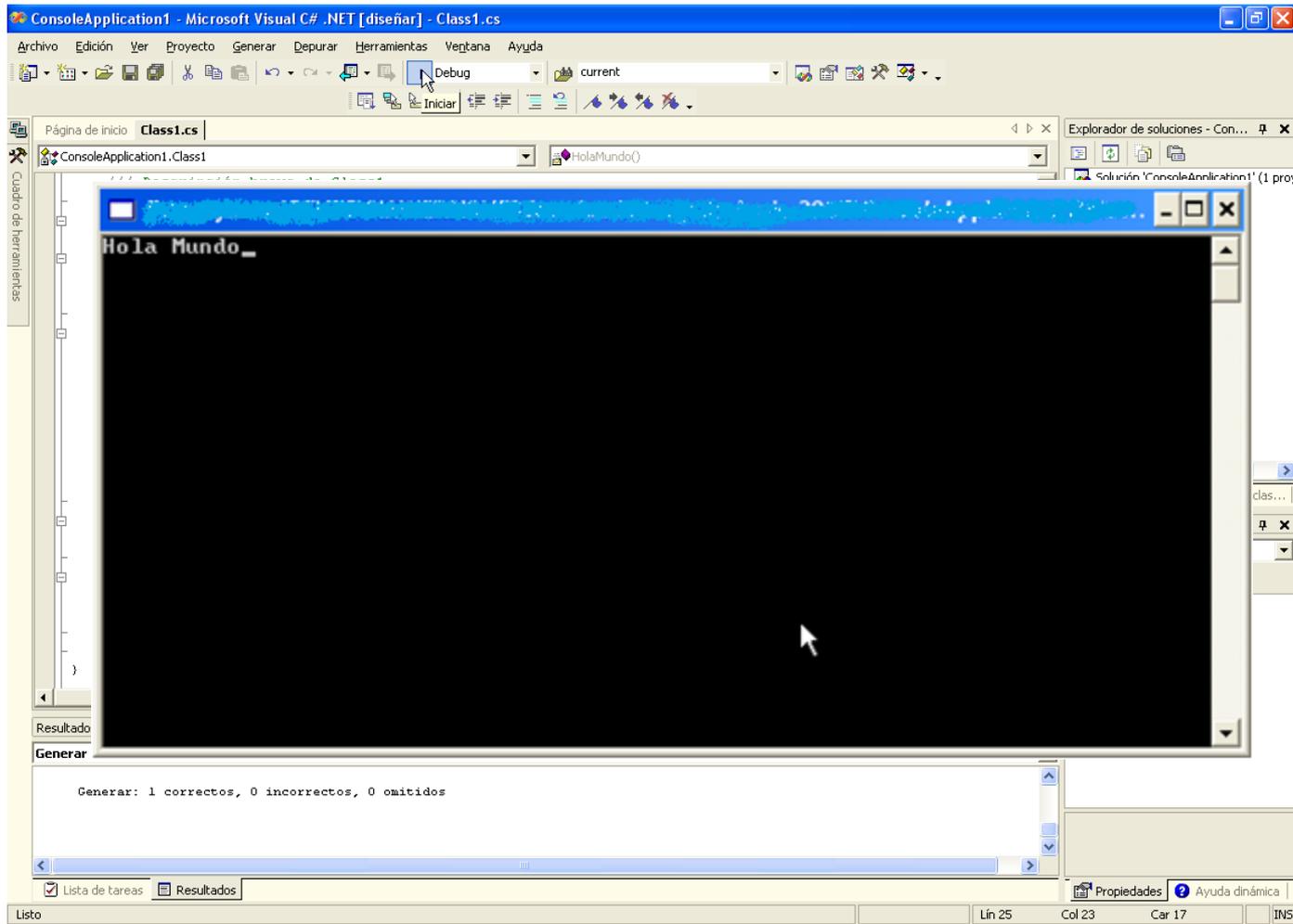
- Code Editor:** Shows the source code for `Class1.cs`. The code includes a class definition for `Class1` with a `Main` method and a `HolaMundo` method.
- Output Window:** Located at the bottom, it shows the build results: `Generar: 1 correctos, 0 incorrectos, 0 omitidos`. This window is circled in orange.
- Solution Explorer:** Shows the project structure for `ConsoleApplication1`, including `App.ico`, `AssemblyInfo.cs`, and `Class1.cs`.
- Properties Window:** Empty.
- Status Bar:** Shows the current cursor position: `Lin 12 Col 23 Car 17 INS`.

# Depuración

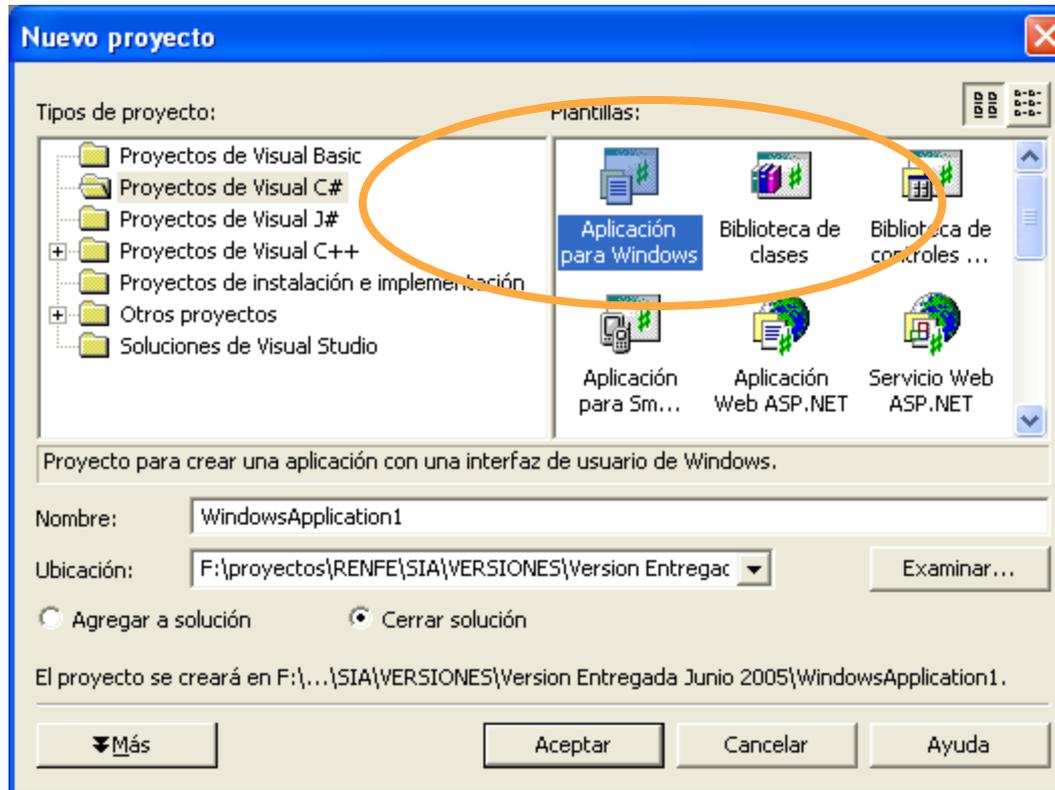
The screenshot shows the Visual Studio IDE with the following components:

- Menu Bar:** Archivo, Edición, Ver, Proyecto, Generar, Depurar, Herramientas, Ventana, Ayuda.
- Toolbar:** Includes icons for file operations, navigation, and a 'Debug' button.
- Code Editor:** Displays the source code for `Class1.cs`. The code includes a class definition for `Class1` with a `Main` method and a `HolaMundo` method. The `Main` method creates a new `Class1` object and calls `HolaMundo`. The `HolaMundo` method writes "Hola Mundo" to the console.
- Output Window:** Shows the result of the compilation: "Generar: 1 correctos, 0 incorrectos, 0 omitidos".
- Task List:** Shows a task to "Actualizar lista de tareas".
- Bottom Status Bar:** Displays "Listo", "Lín 25", "Col 23", "Car 17", and "INS".

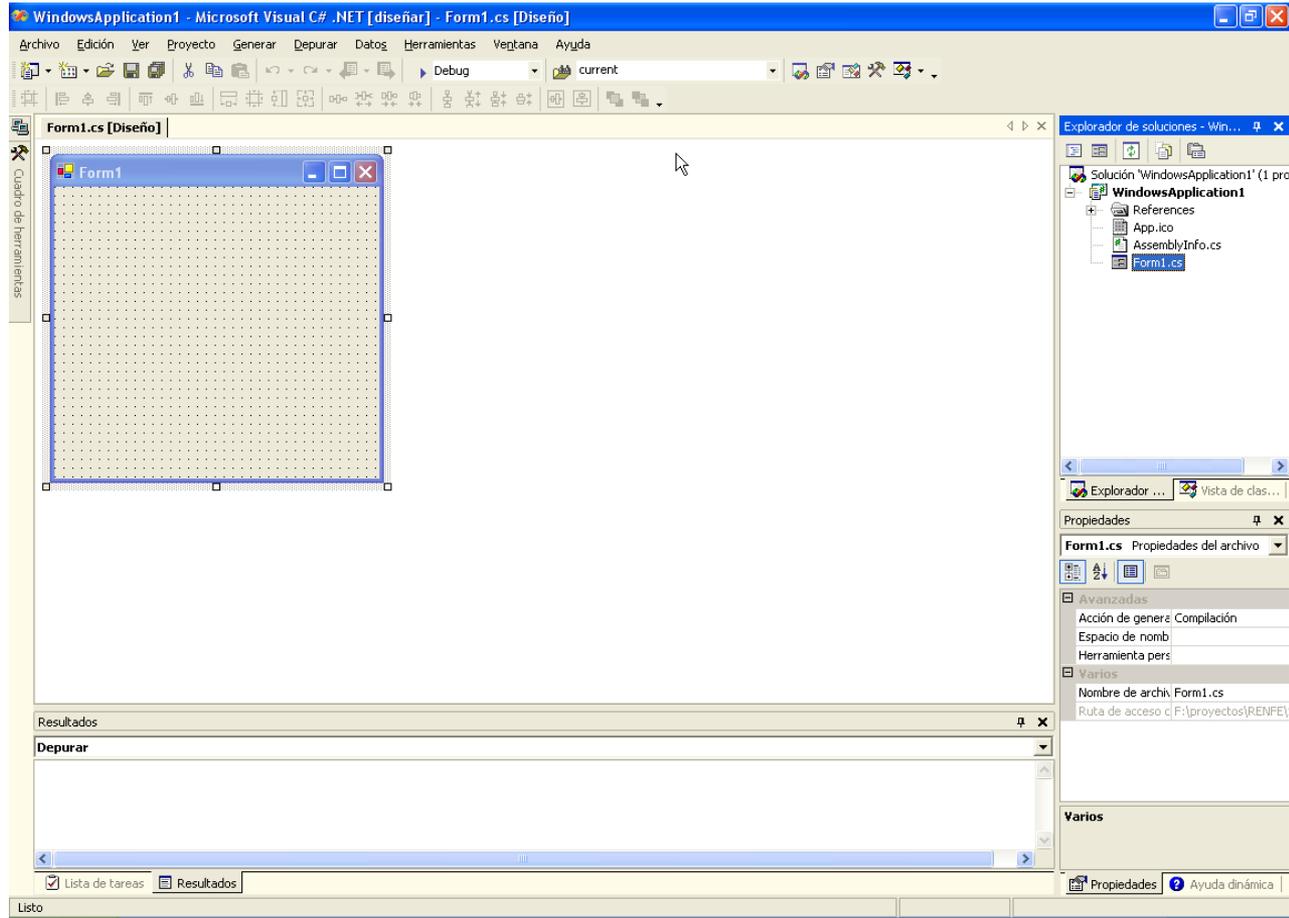
# Resultado de la ejecución



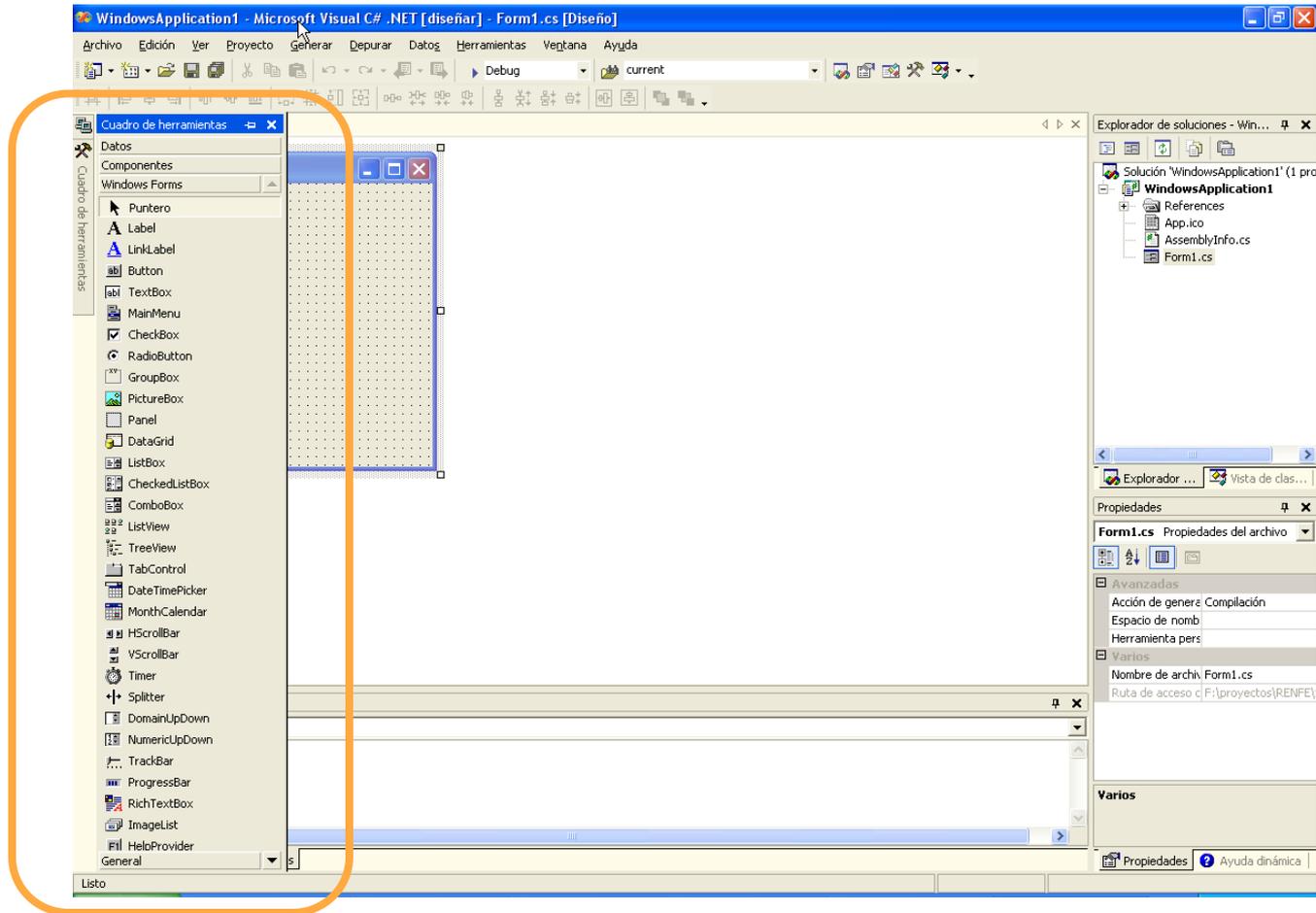
## Ejemplo: Aplicación gráfica



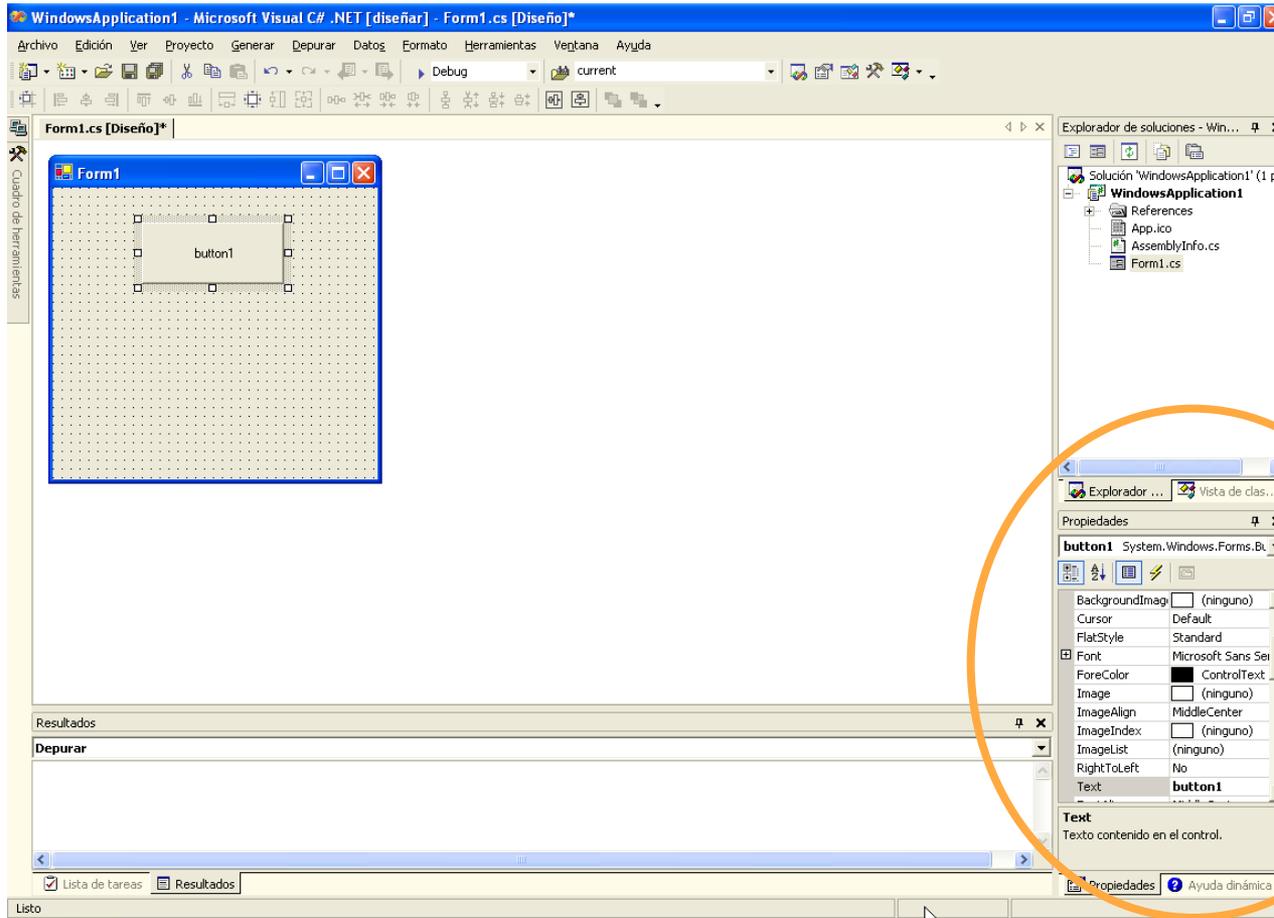
# Creación de la interfaz



# Cuadro de herramientas

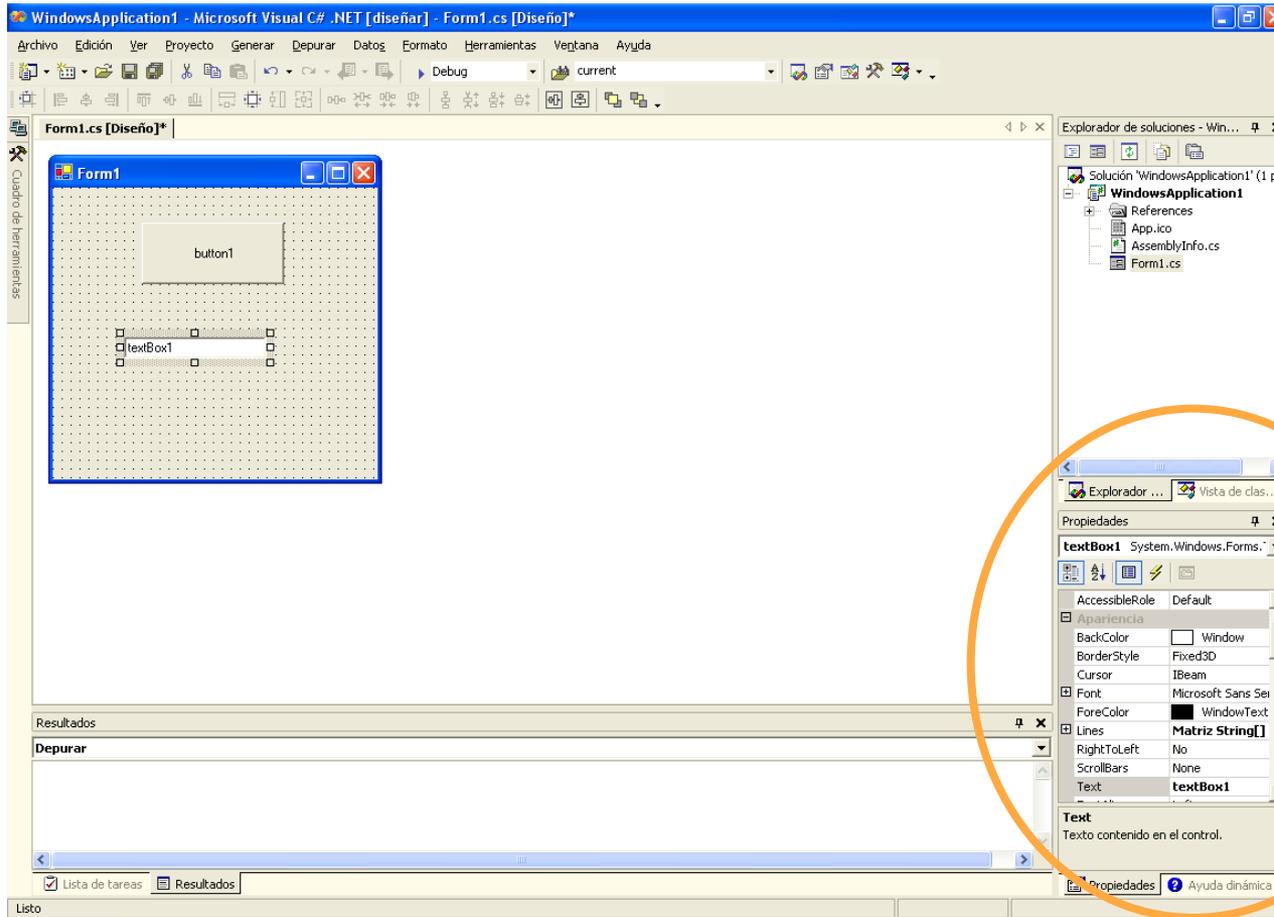


# Inserción de un botón



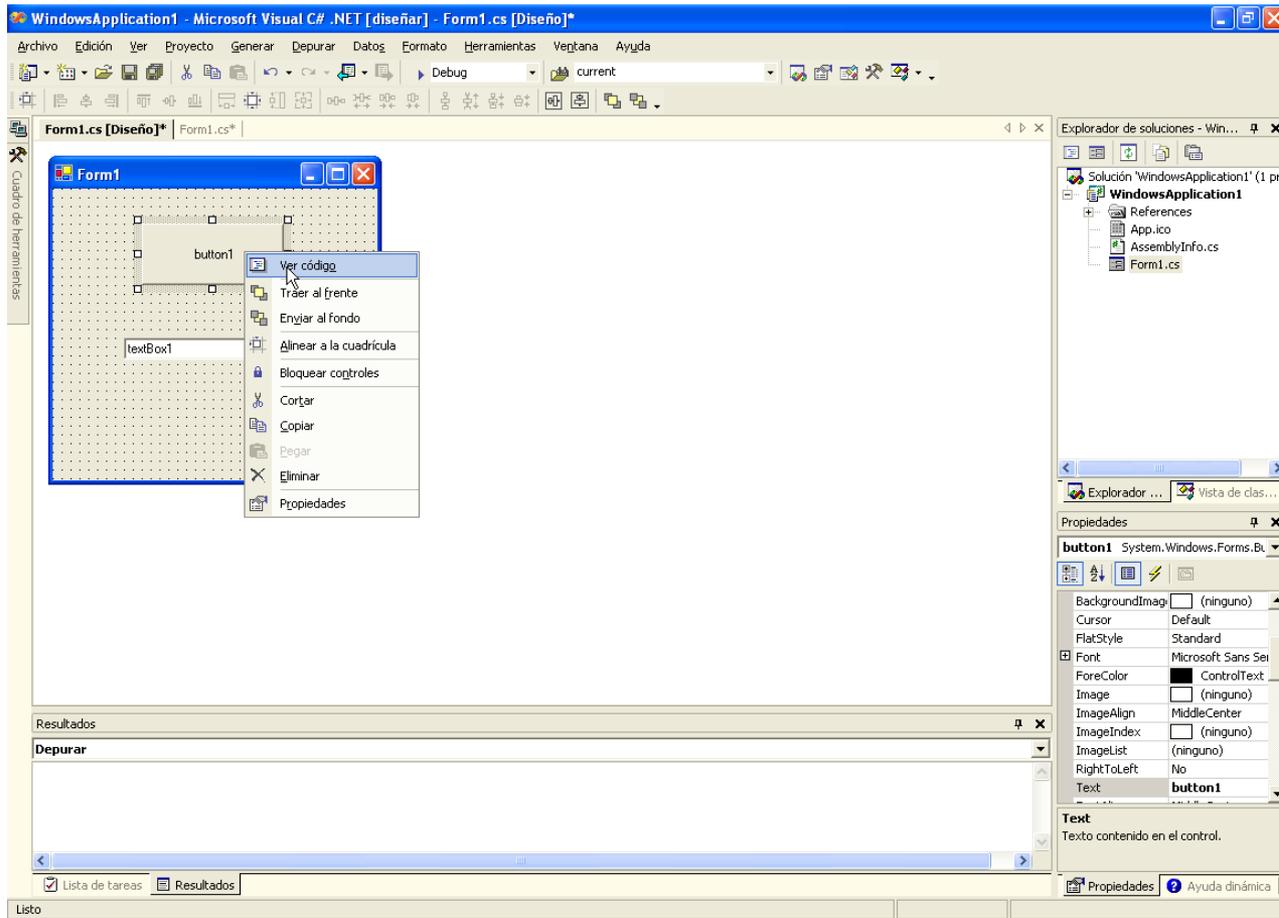
Propiedades  
del botón

# Cuadro de texto



Propiedades  
del cuadro

# Asignación del código al evento



# Programación del evento

The screenshot shows the Microsoft Visual C# .NET IDE with the following code in Form1.cs:

```
using System;
using System.Windows.Forms;

namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Dispose(bool disposing)
        {
            if (components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.textBox1.Location = new System.Drawing.Point(100, 100, 200, 150);
            this.button1.Location = new System.Drawing.Point(100, 200, 150, 250);
            this.textBox1.Text = "Hola Mundo";
            this.button1.Click += new System.EventHandler(this.button1_Click);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.button1);
        }

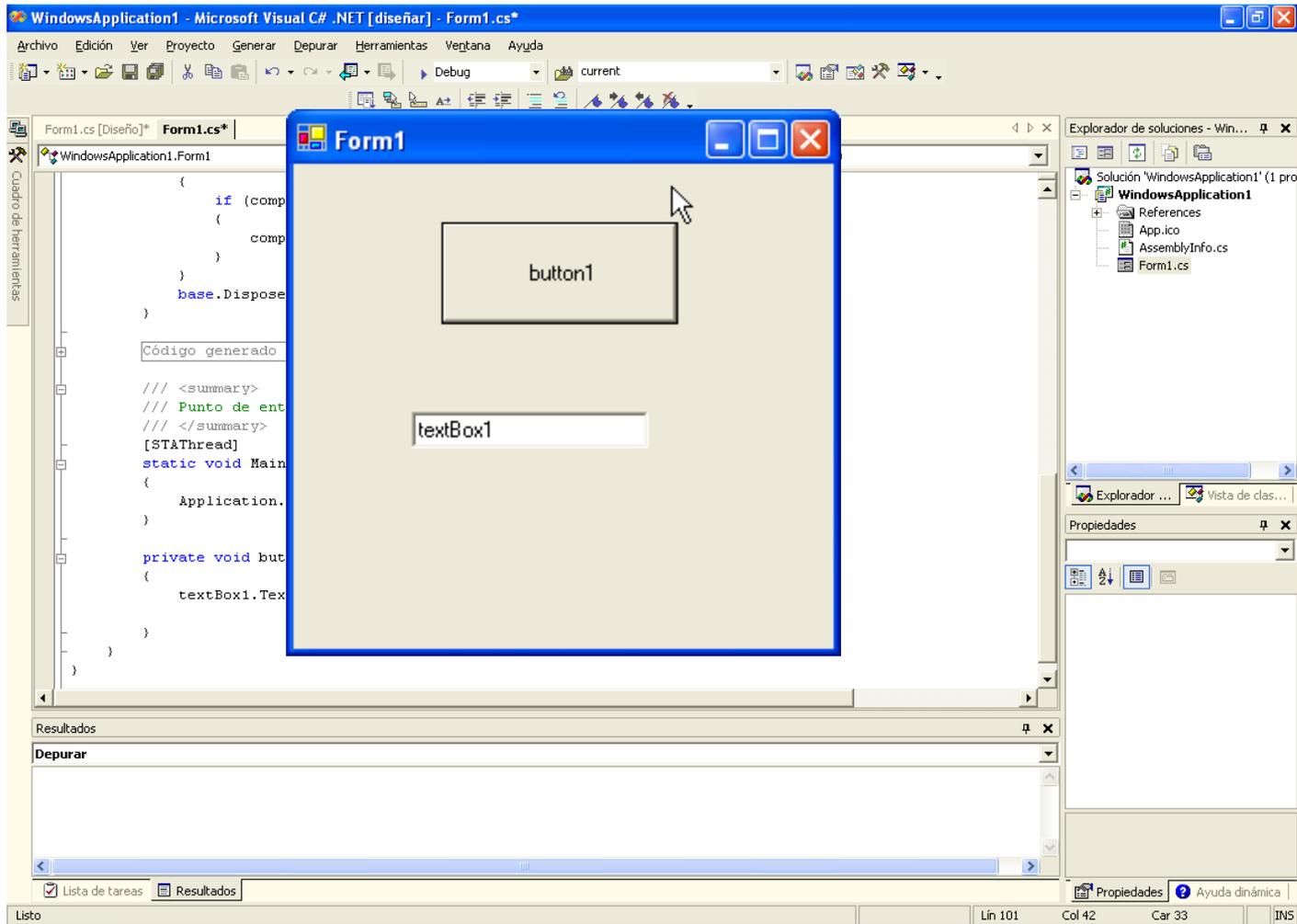
        private void button1_Click(object sender, System.EventArgs e)
        {
            textBox1.Text = "Hola Mundo";
        }
    }
}

/// <summary>
/// Punto de entrada principal de la aplicación.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    textBox1.Text = "Hola Mundo";
}
```

The `button1_Click` method is circled in orange. The IDE interface includes a menu bar (Archivo, Edición, Ver, Proyecto, Generar, Depurar, Herramientas, Ventana, Ayuda), a toolbar, a Solution Explorer on the right showing the project structure, and a Properties window below it.

# Ejecución



# Ejecución

The screenshot shows the Microsoft Visual C# .NET IDE in Design view. The main window, titled "Form1", is running and displays a button labeled "button1" and a text box containing the text "Hola Mundo". The text box is highlighted with an orange oval. The background shows the code editor with the following C# code for Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = "Hola Mundo";
        }
    }
}
```

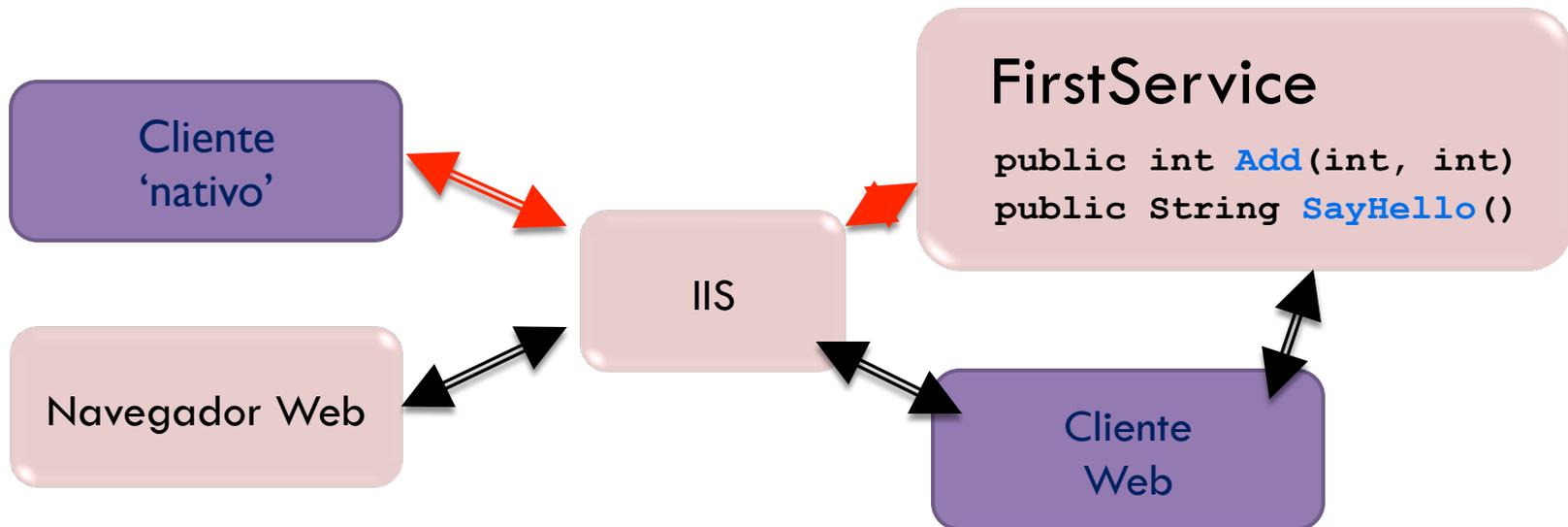
The Solution Explorer on the right shows the project structure for "WindowsApplication1", including references to App.ico, AssemblyInfo.cs, and Form1.cs. The Properties window at the bottom right is empty. The status bar at the bottom indicates "Listo", "Lín 101", "Col 42", "Car 33", and "INS".

# Contenidos

1. Introducción Visual Studio .NET
2. Ejemplo con .NET SDK
3. Ejemplo con Visual Studio 2008

## Servicio Web 'hola mundo'

- Servicio Web **FirstService** con dos métodos, invocado por un **cliente Web** o por un **cliente nativo**



## Servicio Web

1. Codificación del servicio
2. Publicación del servicio
3. Prueba de que el servicio está activo

# 1.- Codificación del servicio Web

- Edición:
  - Escritura del servicio web en C
  - Se guardará en el archivo `c:\temp`  
`\FirstService.asmx`

## FirstService.asmx

```
<%@ WebService language="C#" class="FirstService" %>

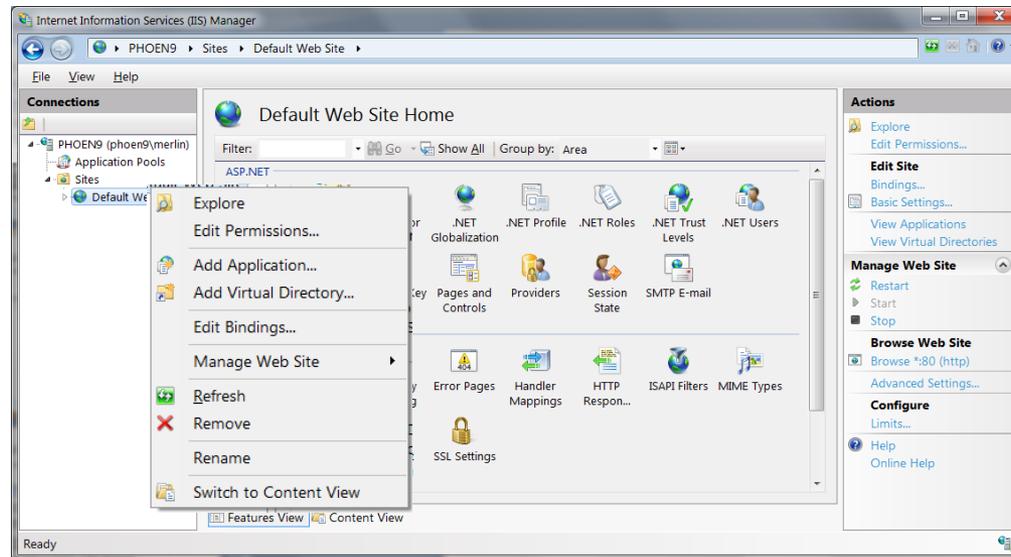
using System;
using System.Web.Services;
using System.Xml.Serialization;

[WebService(Namespace="http://localhost/MyWebServices/")]
public class FirstService : WebService
{
    [WebMethod]
    public int Add(int a, int b)
    {
        return a + b;
    }

    [WebMethod]
    public String SayHello()
    {
        return "Hello World";
    }
}
```

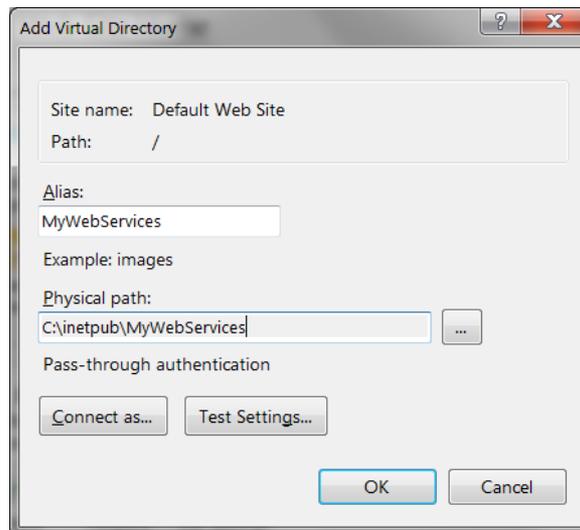
## 2.- Publicación del servicio Web (w7)

- Abrir el gestor de IIS:
  - Windows+R
  - Inetmgr
- Abrir el asistente de directorio virtual:
  - Expandir hasta ver 'Default web site'
  - Botón derecho y seleccionar 'Add Virtual Directory...'



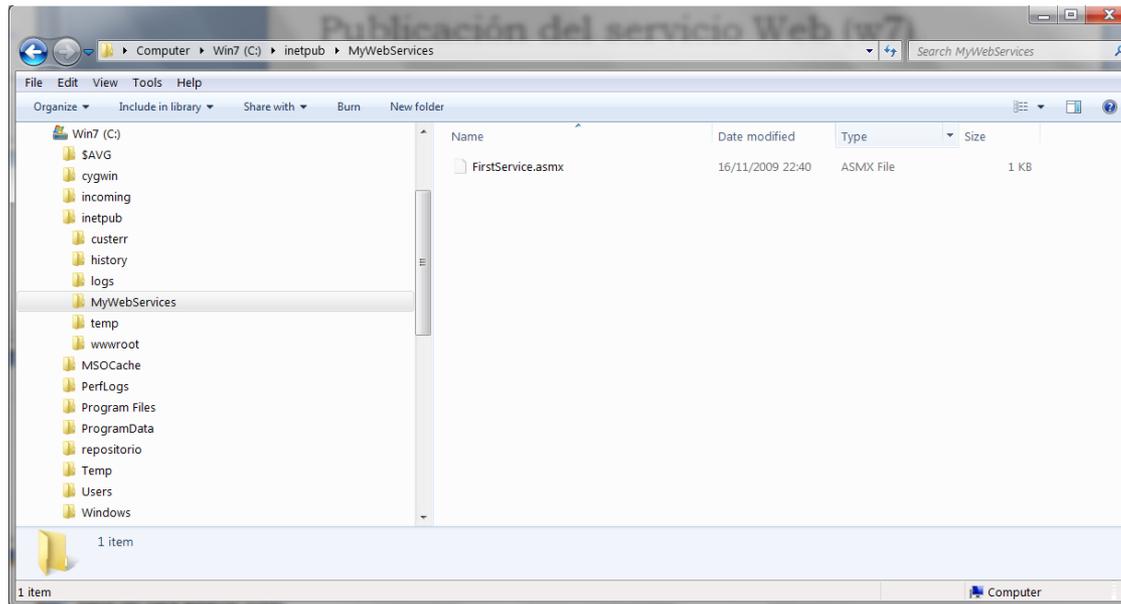
## 2.- Publicación del servicio Web (w7)

- Configurar un nuevo directorio virtual:
  - Indicar el alias (ej.: MyWebServices)
  - Indicar la ruta real completa asociada (ej.: C:\inetpub\MyWebServices)
  - Confirmar con ok



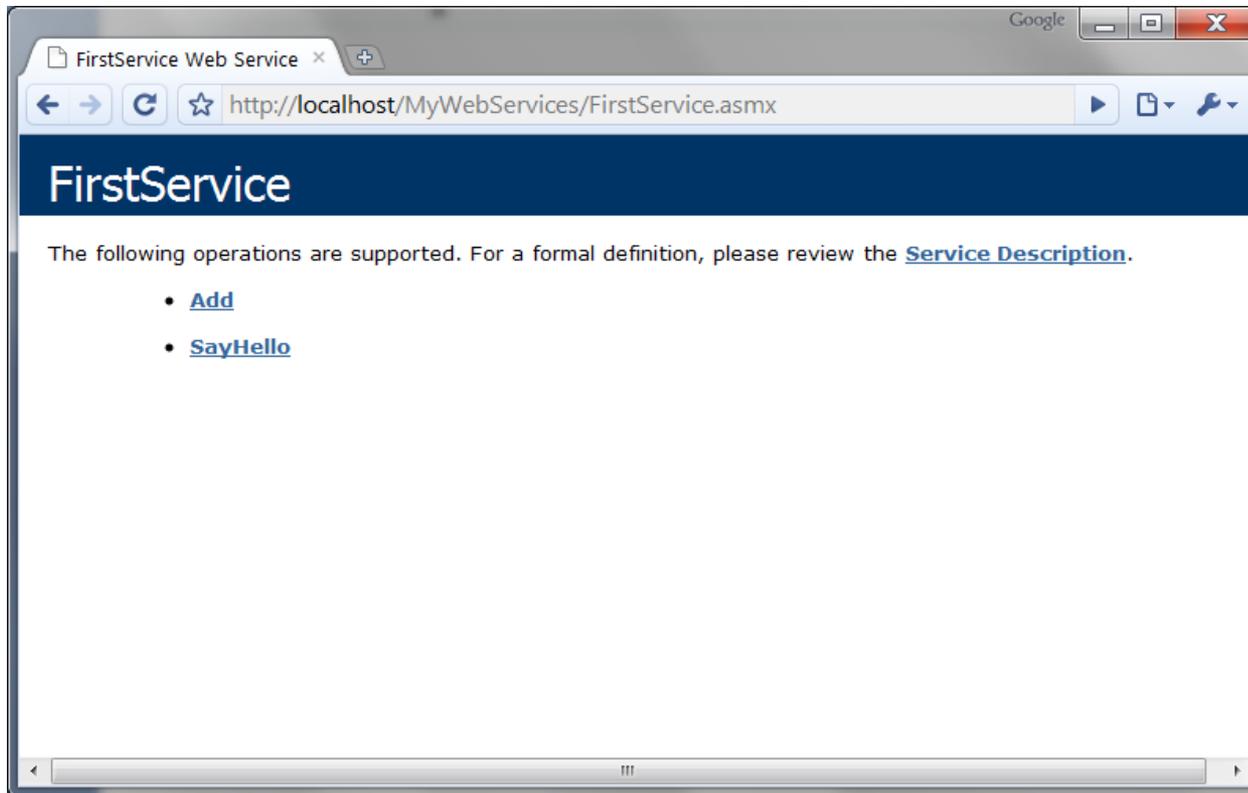
## 2.- Publicación del servicio Web (w7)

- Publicar el servicio:
  - Copiar a C:\inetpub\MyWebServices el archivo c:\temp\FirstService.asmx
  - Confirmar como administrador
  - Es posible usar un directorio virtual por servicio web o un único directorio virtual para todos los servicios web



### 3.- Probar que el servicio está activo

- Abrir la página en navegador web:
  - <http://localhost/MyWebServices/FirstService.asmx>



## 3.- Probar que el servicio está activo

- Probar cada método:
  - Sayhello
  - Add

The screenshot shows a web browser window with the address bar containing `http://localhost/MyWebServices/FirstService.asmx?op=SayHello`. The page title is "FirstService". Below the title, there is a link "Click [here](#) for a complete list of operations." The main heading is "SayHello". Underneath, there is a "Test" section with the instruction "To test the operation using the HTTP POST protocol, click the 'Invoke' button." and an "Invoke" button. Below the test section, there is a "SOAP 1.1" section with the text "The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values." and a code block containing a sample SOAP request and response.

```
POST /MyWebServices/FirstService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://localhost/MyWebServices/SayHello"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SayHello xmlns="http://localhost/MyWebServices/" />
  </soap:Body>
</soap:Envelope>
```

The screenshot shows a web browser window with the address bar containing `http://localhost/MyWebServices/FirstService.asmx?op=Add`. The page title is "FirstService". Below the title, there is a link "Click [here](#) for a complete list of operations." The main heading is "Add". Underneath, there is a "Test" section with the instruction "To test the operation using the HTTP POST protocol, click the 'Invoke' button." and an "Invoke" button. Below the test section, there is a "SOAP 1.1" section with the text "The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values." and a code block containing a sample SOAP request and response.

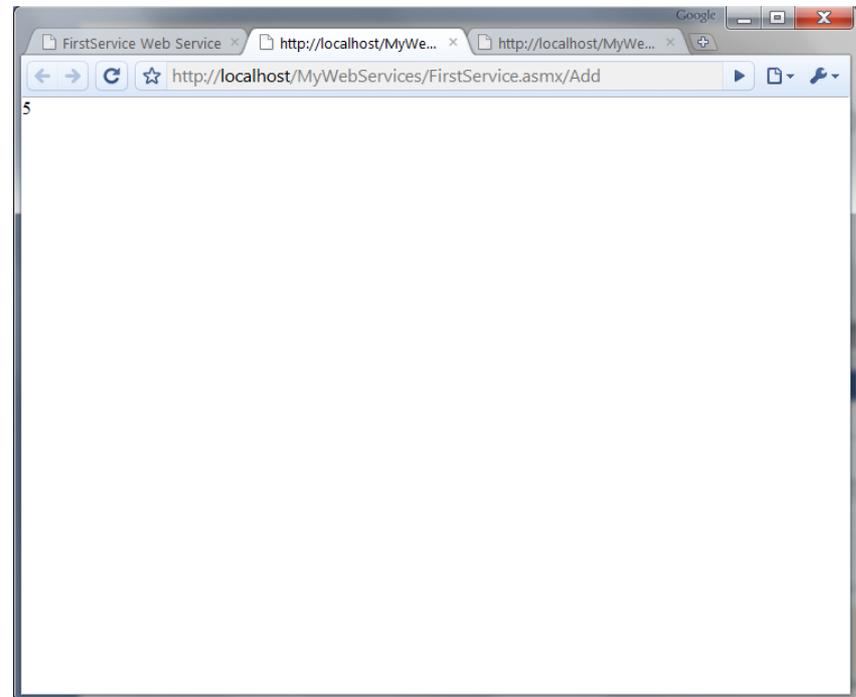
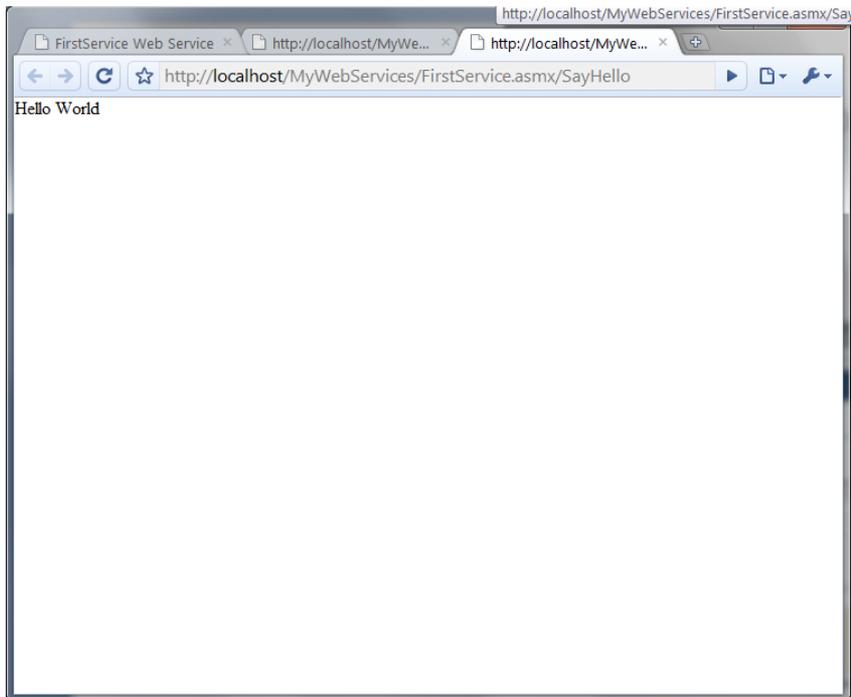
Parameter	Value
a:	<input type="text"/>
b:	<input type="text"/>

```
POST /MyWebServices/FirstService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://localhost/MyWebServices/Add"

<?xml version="1.0" encoding="utf-8"?>
```

### 3.- Probar que el servicio está activo

- Probar cada método:
  - sayhello
  - Add



# Consumidor del servicio Web

## Aplicación Web

1. Creación del proxy
2. Creación del cliente Web
3. Publicación del cliente Web
4. Ejecución de la aplicación Web

# 1.- Creación del proxy

## □ Preprocesado:

```
C:\temp> cmd.exe /V:ON /E:ON /K "C:\Program Files\Microsoft SDKs\→  
→ windows\v6.1\bin\setenv.cmd" /Release  
C:\temp> WSDL http://localhost/MyWebServices/FirstService.asmx?WSDL
```

## □ Compilación:

```
C:\Temp\> csc /t:library FirstService.cs
```

## □ Despliegue:

- Copiar de `c:\temp\FirstService.dll`  
a `C:\inetpub\MyWebServices\bin`

## 2.- Creación del cliente Web

- Edición:
  - ▣ Escritura del consumidor web en ASP.NET
  - ▣ Se guardará en el archivo `c:\temp\WebApp.aspx`

## WebApp.aspx (1/3)

```
<%@ Page
Language="C#" CompilerOptions=' /R:"C:\inetpub\MyWebServices\bin
\FirstService.dll"'
%>

<script runat="server">
void runSrvice_Click(Object sender, EventArgs e)
{
    FirstService mySvc = new FirstService();
    Label1.Text = mySvc.SayHello();
    Label2.Text = mySvc.Add(Int32.Parse(txtNum1.Text),
        Int32.Parse(txtNum2.Text)).ToString();
}
</script>
<html>
<head>
</head>
<body>
```

## WebApp.aspx (2/3)

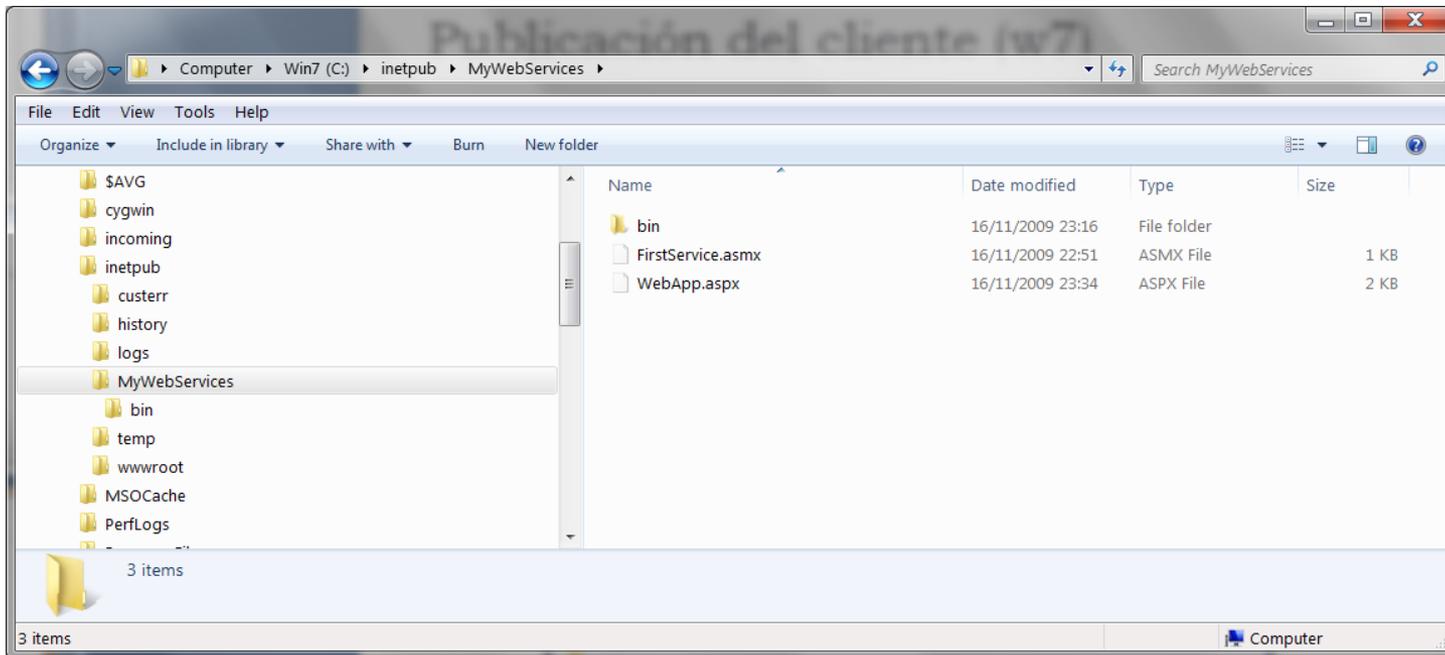
```
<form runat="server">
  <p>
    <em>First Number to Add </em>:
    <asp:TextBox id="txtNum1" runat="server"
      Width="43px">4</asp:TextBox>
  </p>
  <p>
    <em>Second Number To Add </em>:
    <asp:TextBox id="txtNum2" runat="server"
      Width="44px">5</asp:TextBox>
  </p>
  <p>
    <strong><u>Web Service Result -</u></strong>
  </p>
```

## WebApp.aspx (3/3)

```
<p>
  <em>Hello world Service</em> :
  <asp:Label id="Label1" runat="server"
    Font-Underline="True">Label</asp:Label>
</p>
<p>
  <em>Add Service</em> :
  & <asp:Label id="Label2" runat="server"
    Font-Underline="True">Label</asp:Label>
</p>
<p align="left">
  <asp:Button id="runSrvice" onclick="runSrvice_Click"
    runat="server" Text="Execute"></asp:Button>
</p>
</form>
</body>
</html>
```

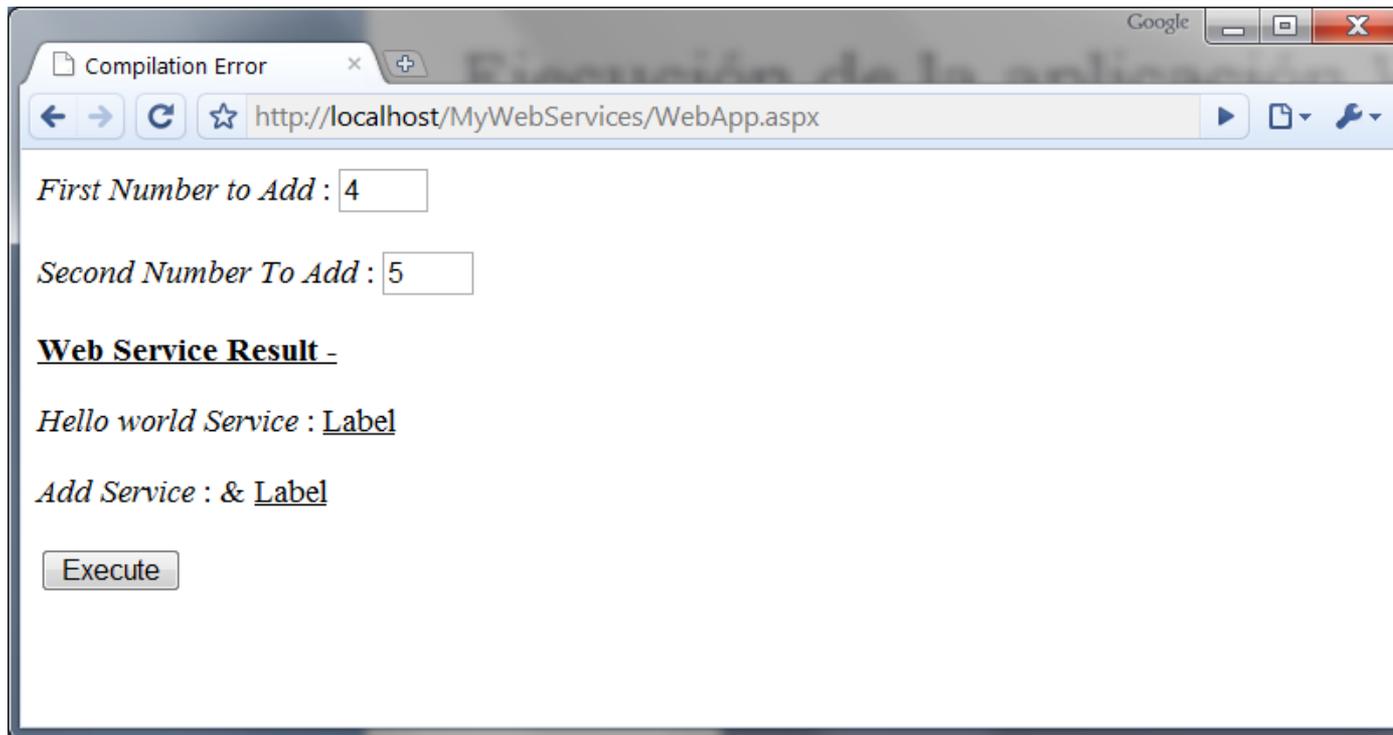
## 3.- Publicación del cliente (w7)

- Publicar el servicio:
  - Copiar a `C:\inetpub\MywebServices` el archivo `c:\temp\WebApp.aspx`
  - Confirmar como administrador



## 4.- Ejecución de la aplicación Web

- Abrir la página en navegador Web:
  - <http://localhost/MyWebServices/WebApp.aspx>



# Consumidor del servicio Web

## Aplicación nativa

1. Creación del proxy
2. Creación del cliente nativo
3. Ejecución de la aplicación nativa

# 1.- Creación del proxy

## □ Preprocesado:

```
C:\temp> cmd.exe /V:ON /E:ON /K "C:\Program Files\Microsoft SDKs\→  
→ windows\v6.1\bin\setenv.cmd" /Release  
C:\temp> WSDL http://localhost/MywebServices/FirstService.asmx?WSDL
```

## □ Compilación:

```
C:\Temp\> csc /t:library FirstService.cs
```

## 2.- Creación del cliente nativo

### □ Código:

```
using System;
using System.IO;

namespace SvcConsumer {
    class SvcEater {
        public static void Main(String[] args) {
            FirstService mySvc = new FirstService();
            Console.WriteLine("Calling Hello World Service: " + mySvc.SayHello());
            Console.WriteLine("Calling Add(2, 3) Service: " + mySvc.Add(2, 3).ToString());
        }
    }
}
```

WinApp.cs

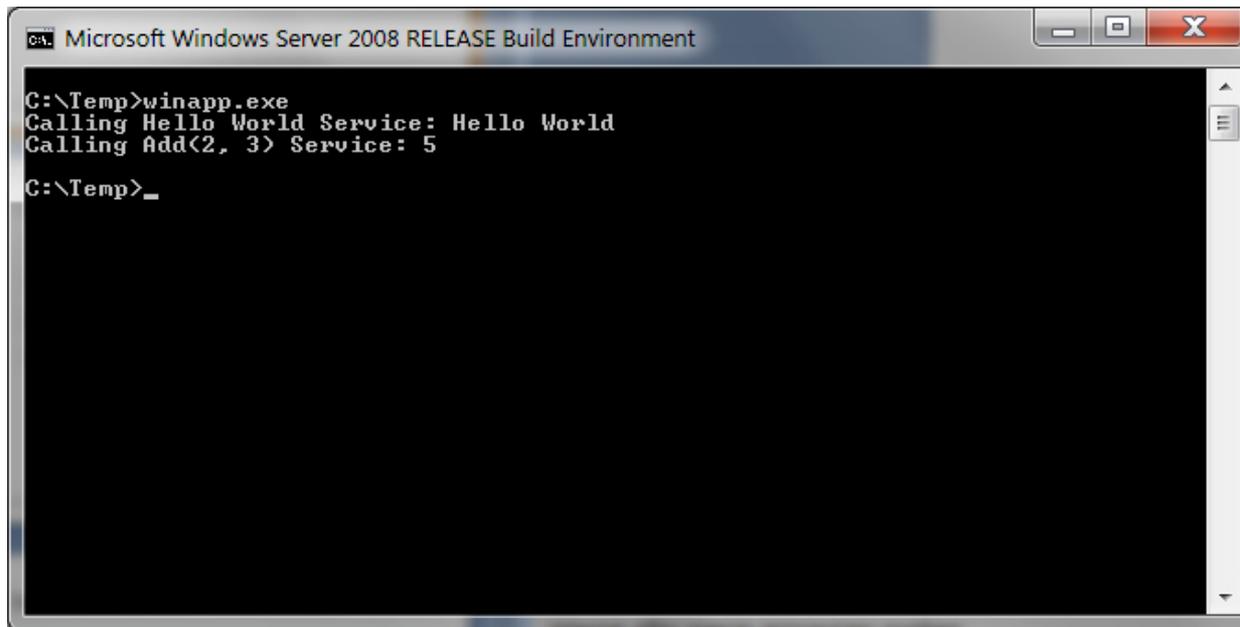
### □ Compilación:

```
C:\Temp> csc /r:FirstService.dll WinApp.cs
```

## 3.- Ejecución del cliente nativo

- Ejecutar como cualquier otra aplicación:

```
C:\Temp> WinApp.exe
```



```
Microsoft Windows Server 2008 RELEASE Build Environment
C:\Temp>winapp.exe
Calling Hello World Service: Hello World
Calling Add(2, 3) Service: 5
C:\Temp>_
```

# Contenidos

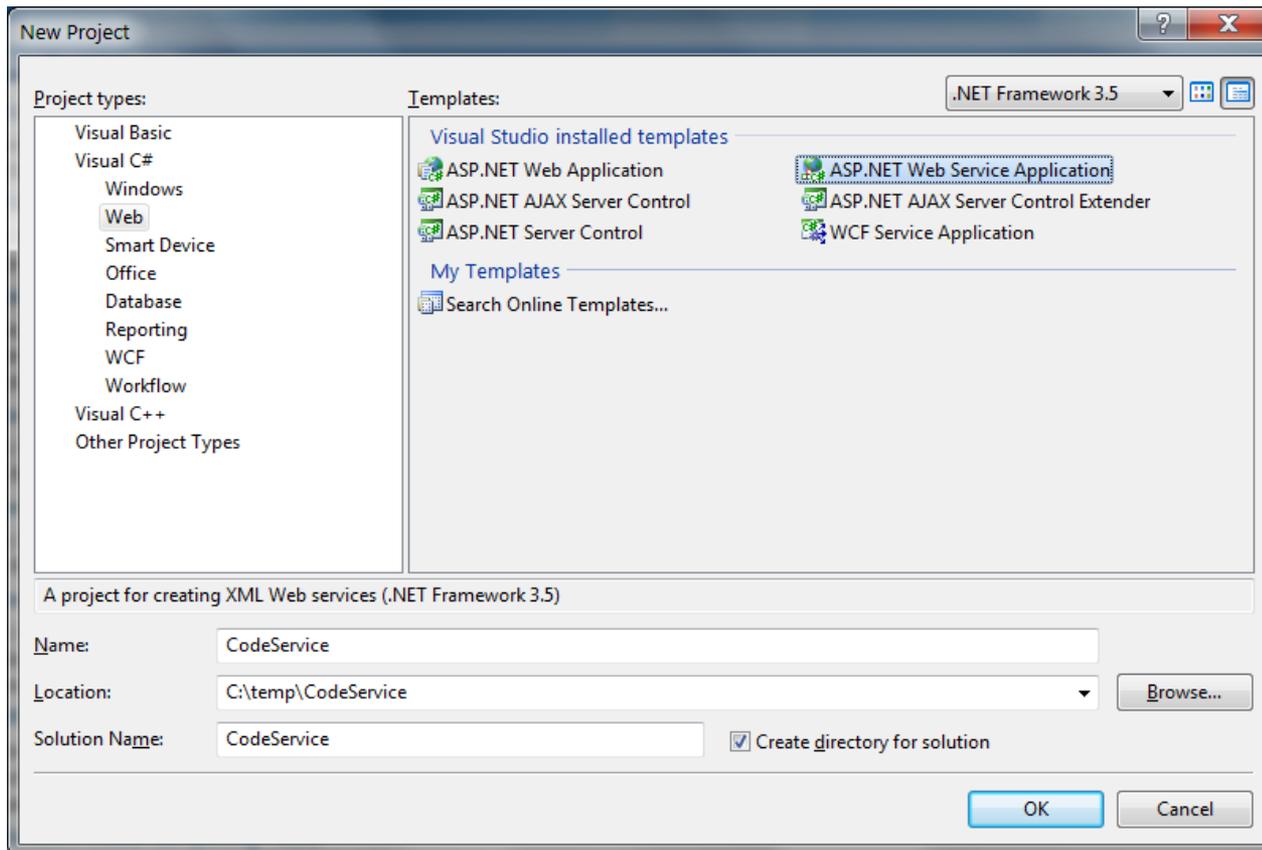
1. Introducción Visual Studio .NET
2. Ejemplo con .NET SDK
3. Ejemplo con Visual Studio 2008

# Servicio Web con Visual Studio

1. Creación del proyecto
2. Codificación del servicio Web
3. Comprobación de que está activo

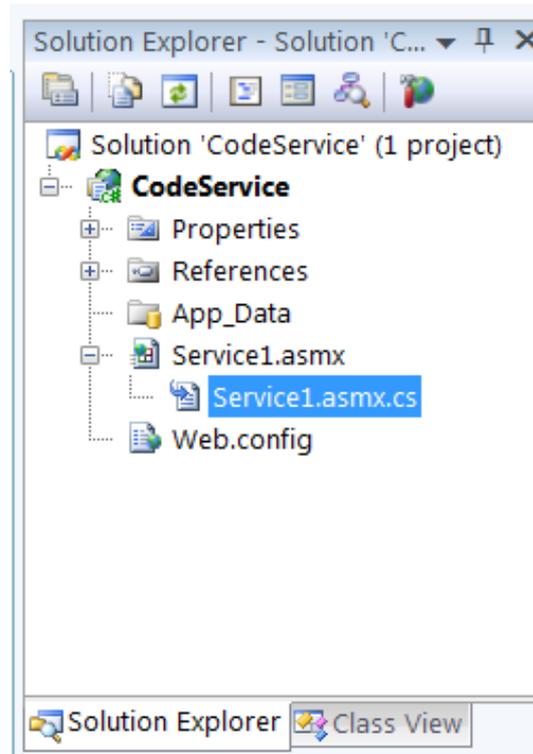
# 1.- Creación del proyecto

- Nombre del proyecto: CodeService



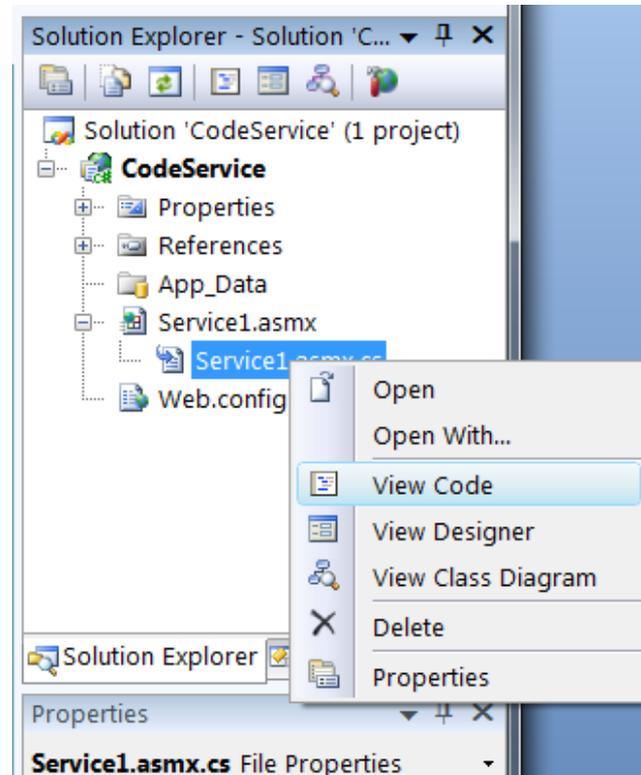
## 1.- Creación del proyecto

- Posible contenido inicial del proyecto:



# 1.- Creación del proyecto

- Pasar a *View Code*:



## 2.- Codificación del servicio Web

### □ Cambiar:

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

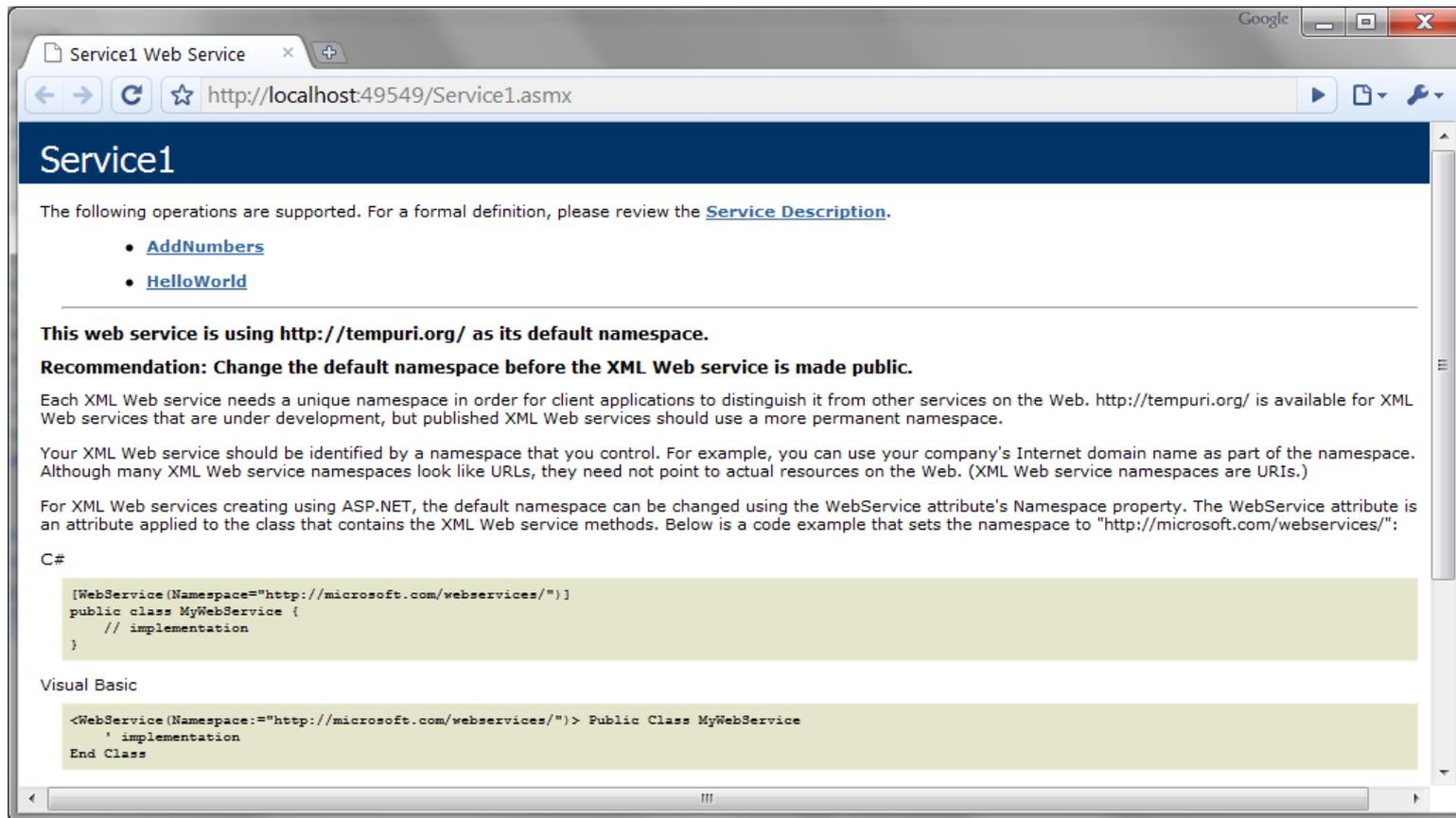
### □ Por:

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

```
[WebMethod]
public decimal AddNumbers(decimal lnNumber1, decimal lnNumber2) {
    return lnNumber1 + lnNumber2;
}
```

## 3.- Comprobación de que está activo

- Ejecutar el proyecto (tecla F5).



Service1

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [AddNumbers](#)
- [HelloWorld](#)

**This web service is using <http://tempuri.org/> as its default namespace.**

**Recommendation: Change the default namespace before the XML Web service is made public.**

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. <http://tempuri.org/> is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "<http://microsoft.com/webservices/>":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

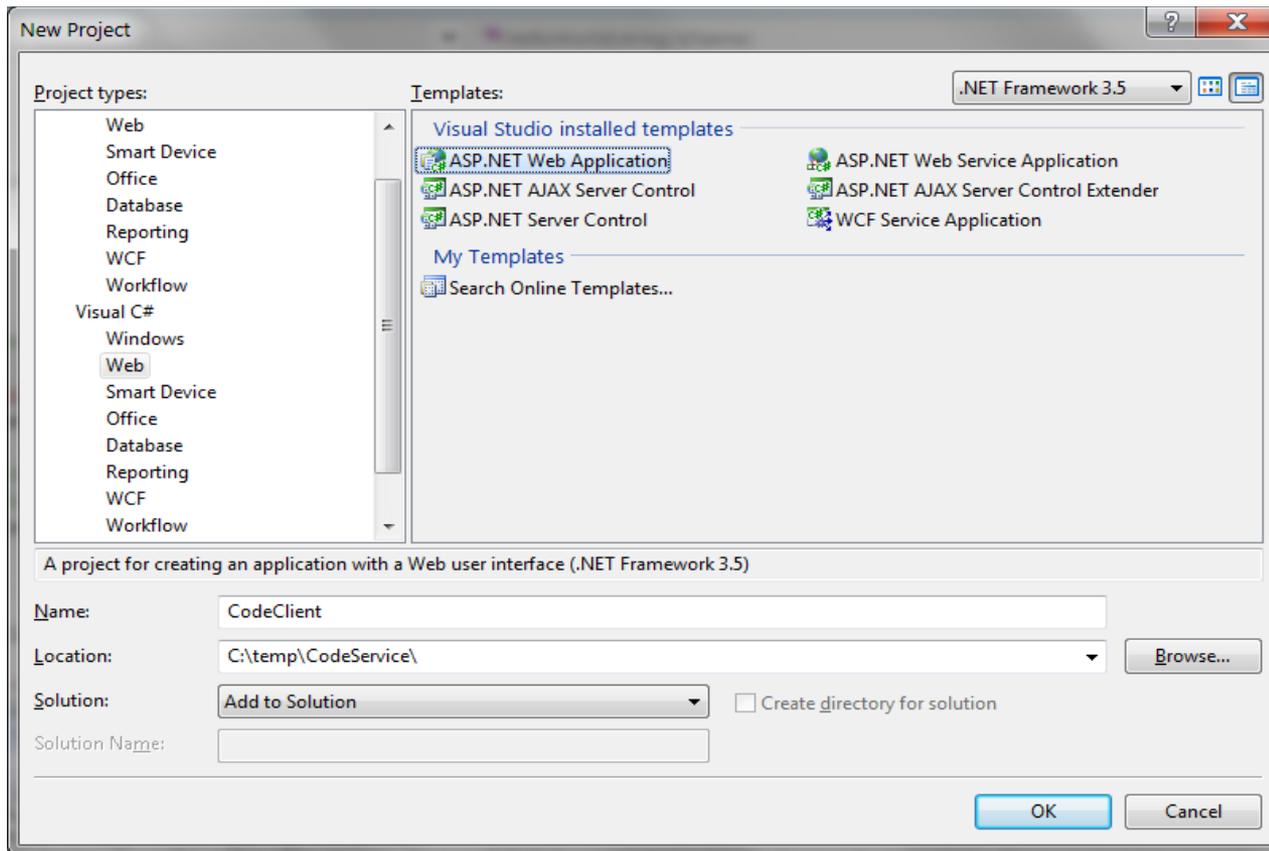
```
<WebService(Namespace="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

# Cliente con Visual Studio

1. Creación del proyecto
2. Codificación del cliente
3. Ejecución del cliente

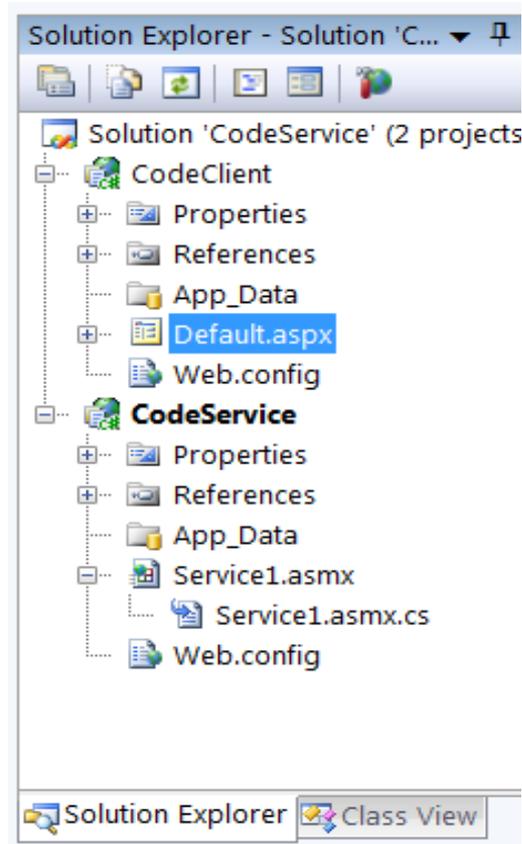
# 1.- Creación del proyecto

- Nombre del proyecto: **CodeClient**



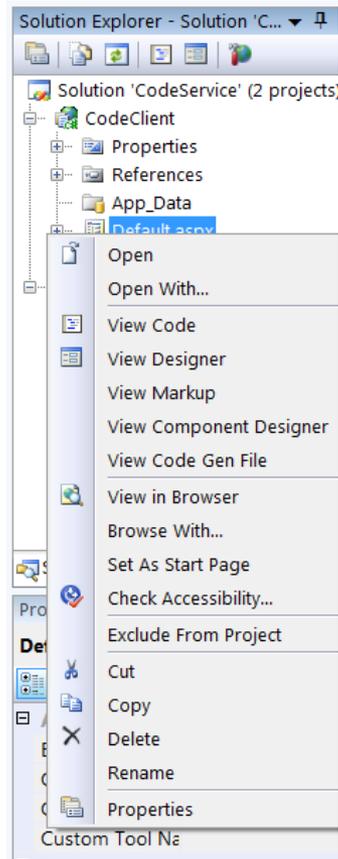
## 1.- Creación del proyecto

- Posible contenido inicial del proyecto:



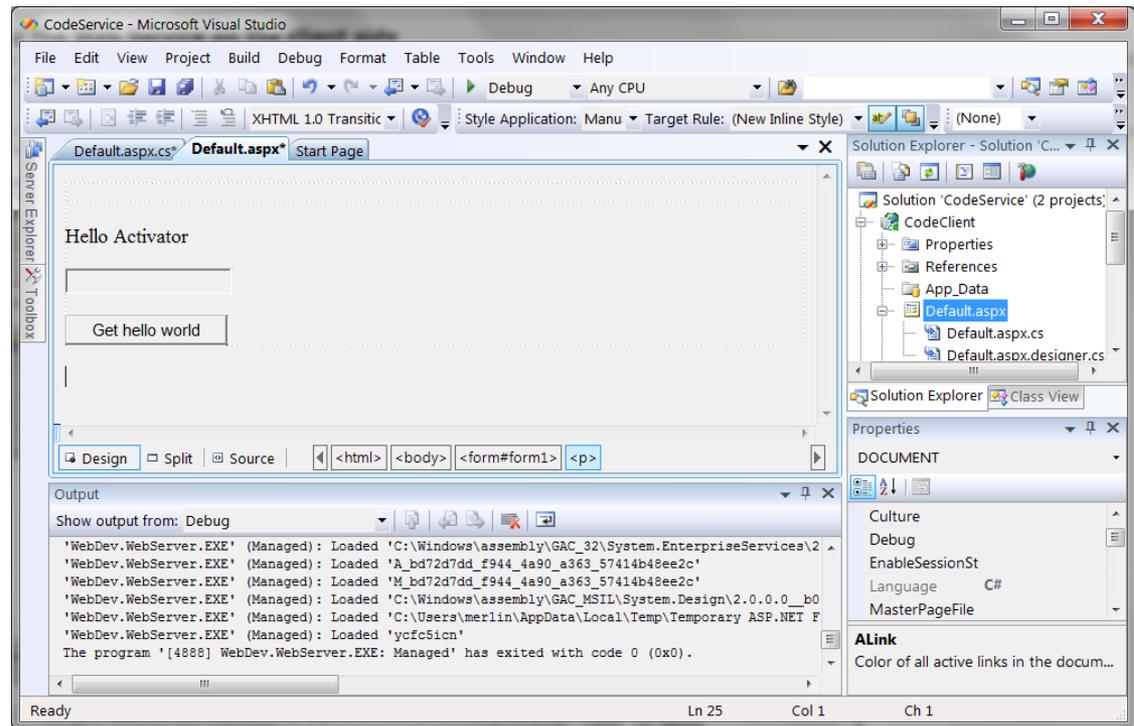
# 1.- Creación del proyecto

- Pasar a *View Designer*:



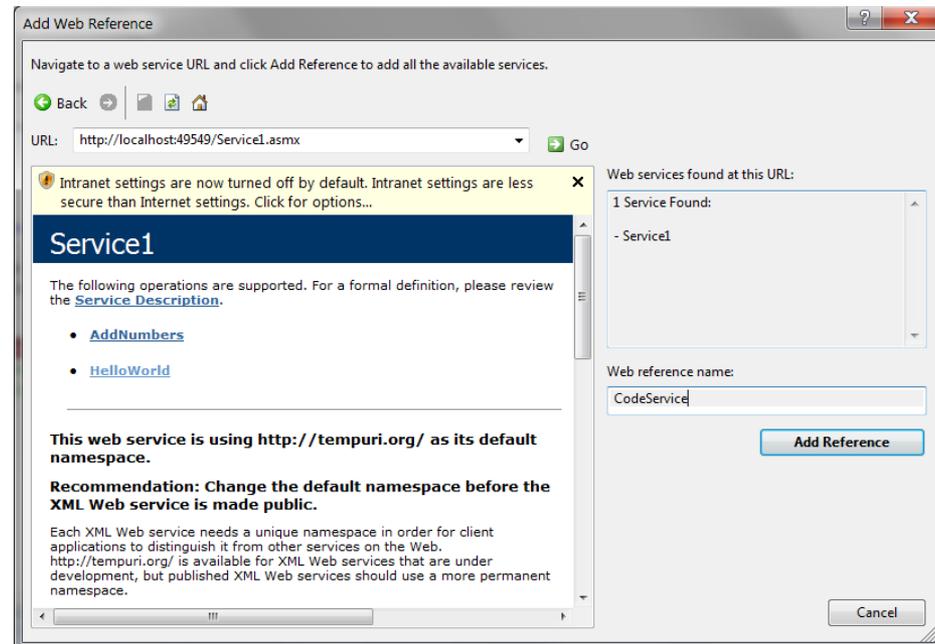
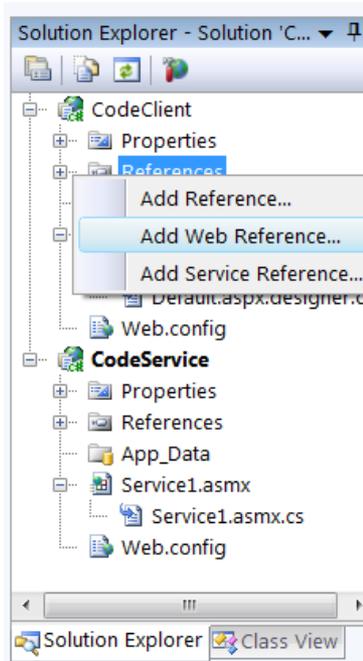
## 2.- Codificación del proyecto

- Diseñar la interfaz:
  - Añadir una caja de texto de nombre **TextBox1**
  - Añadir un botón de nombre **Button1**



## 2.- Codificación del proyecto

- Añadir referencia web:
  - Buscar el servicio en la solución



## 2.- Codificación del proyecto

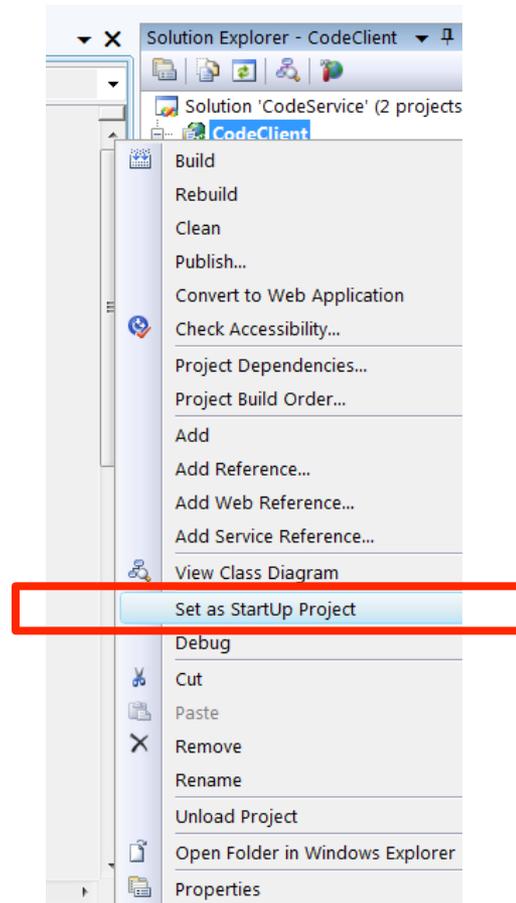
- Añadir la invocación al botón:

```
protected void Button1_Click(object sender, EventArgs e)
{
    CodeService.Service1 oService = new CodeService.Service1();

    TextBox1.Text = oService.HelloWorld("ARCOS.UC3M");
}
```

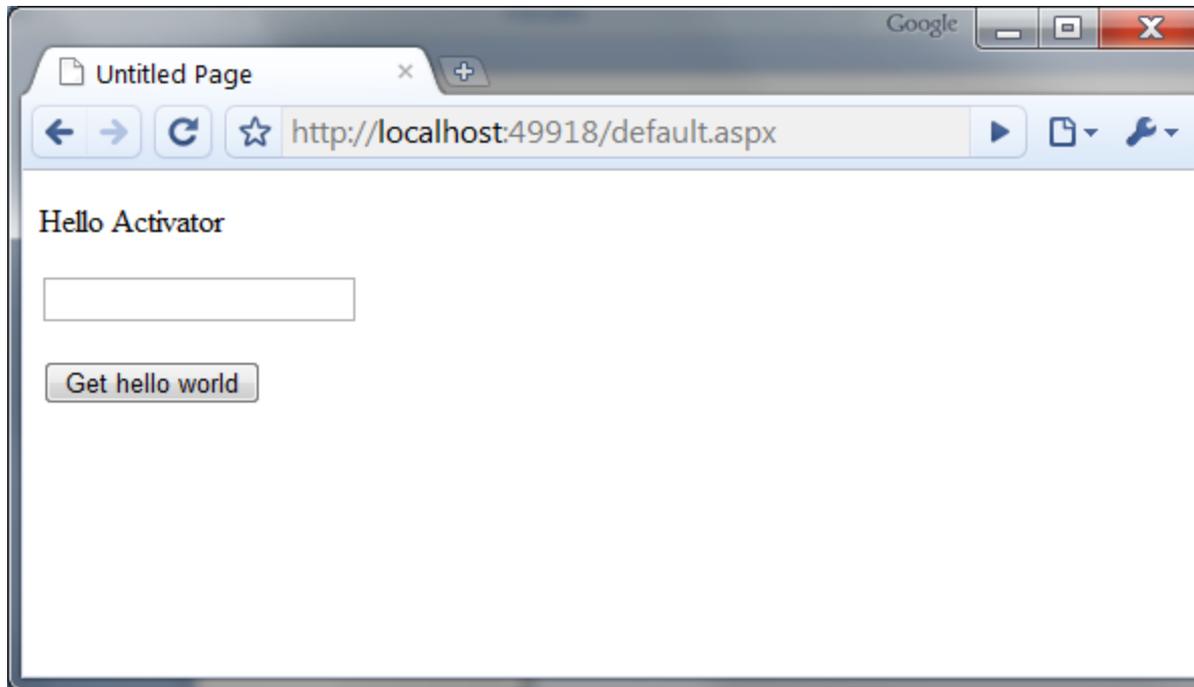
### 3.- Comprobación de que está activo

- Establecer **codeClient** como proyecto de trabajo



### 3.- Comprobación de que está activo

- Ejecutar el proyecto (tecla F5).



### 3.- Comprobación de que está activo

- Ejecutar el proyecto (tecla F5).

