APLICACIONES WEB CON ASP.NET

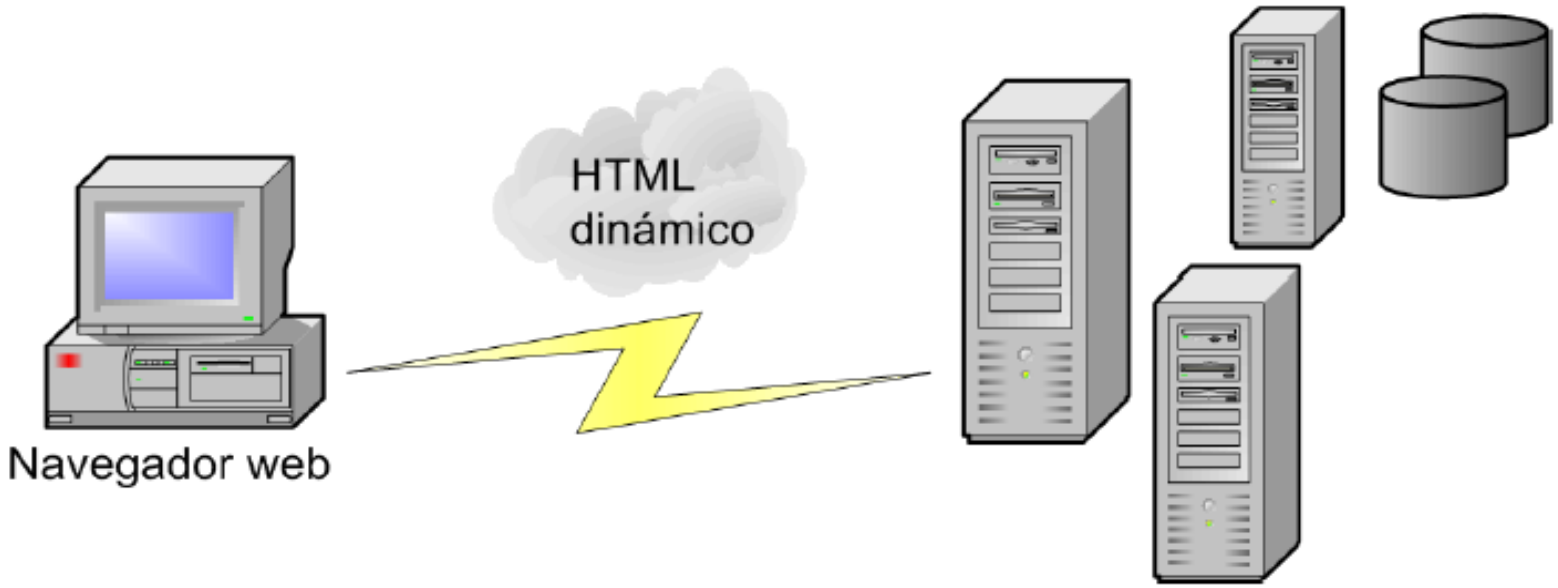
EVOLUCIÓN DE LAS APLICACIONES WEB

- HTML ESTÁTICO

HTML estático: Configuración típica de una "aplicación web" que se limita a ofrecer la información almacenada en páginas HTML a las que el usuario final accede desde su navegador web utilizando el protocolo HTTP.

EVOLUCIÓN DE LAS APLICACIONES WEB

- HTML DINÁMICO

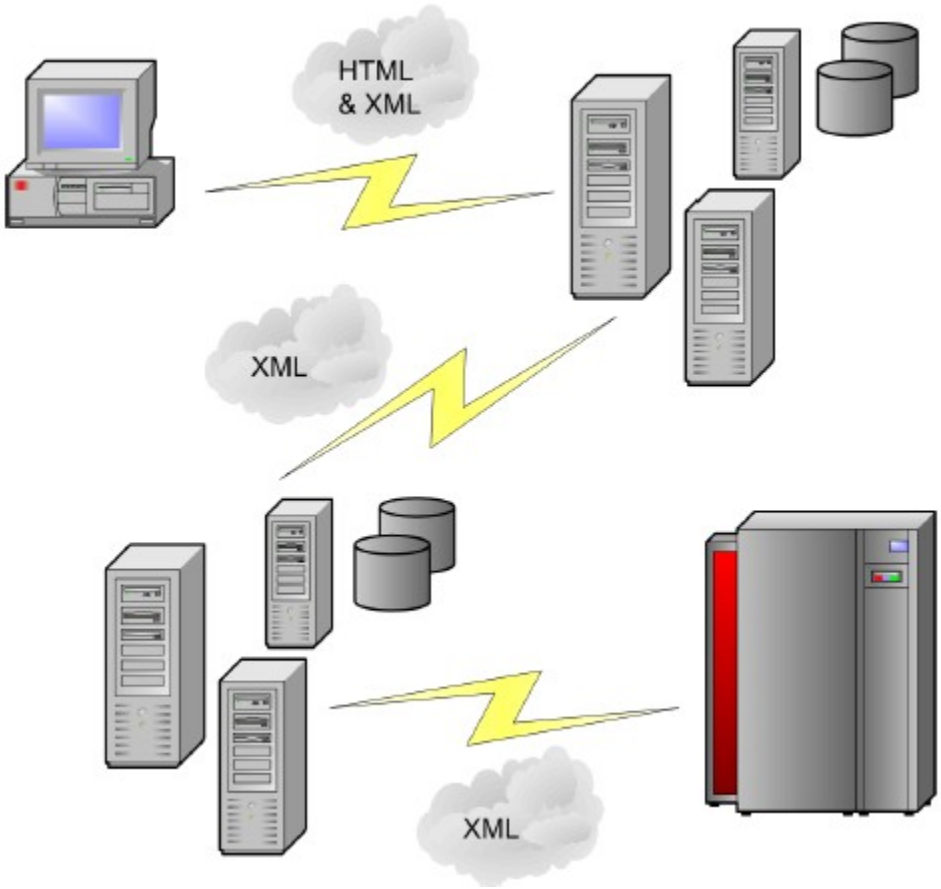


Aplicaciones web: El contenido que se le muestra al usuario se genera dinámicamente para cada solicitud proveniente del navegador web instalado en la máquina cliente.

EVOLUCIÓN DE LAS APLICACIONES WEB

- Servicios Web
  - Los servicios web, básicamente, establecen un lenguaje común mediante el cual distintos sistemas puedan comunicarse entre sí y, de esta forma, facilitan la construcción de sistemas distribuidos heterogéneos.

Servicios web



La lógica de la aplicación se distribuye.
El intercambio de mensajes en formato XML y el uso de protocolos estándar de Internet nos permiten tener conectadas las distintas partes de una aplicación, aunque ésta haya de funcionar en un sistema distribuido heterogéneo.

DESARROLLO DE APLICACIONES PARA INTERNET

- A grandes rasgos, podemos diferenciar dos grandes grupos de aplicaciones web en función de si la lógica de la aplicación se ejecuta en el cliente o en el servidor.

EN EL CLIENTE

- En principio, todo el software asociado a una aplicación web se puede desarrollar de forma que el trabajo lo realice el servidor y el navegador instalado en la máquina cliente se limite a mostrar páginas HTML generadas en el servidor.
- Sin embargo las limitaciones del formato HTML para construir interfaces de usuario (algo para lo que nunca fue diseñado) ha propiciado la aparición de numerosas tecnologías que permiten ejecutar código en la máquina del cliente

EN EL CLIENTE

- Algunas de las herramientas y tecnologías que se suelen utilizar para ejecutar parte de la aplicación web en la máquina del propio cliente son las siguientes:
  - HTML dinámico y Javascript
  - Controles ActiveX (Microsoft)
  - Applets (Java)
  - Plug-ins específicos

EN EL SERVIDOR

- Independientemente de que utilicemos o no las tecnologías comentadas anteriormente, la funcionalidad de las aplicaciones web usualmente la tendremos que implementar en el lado del servidor.

En el servidor

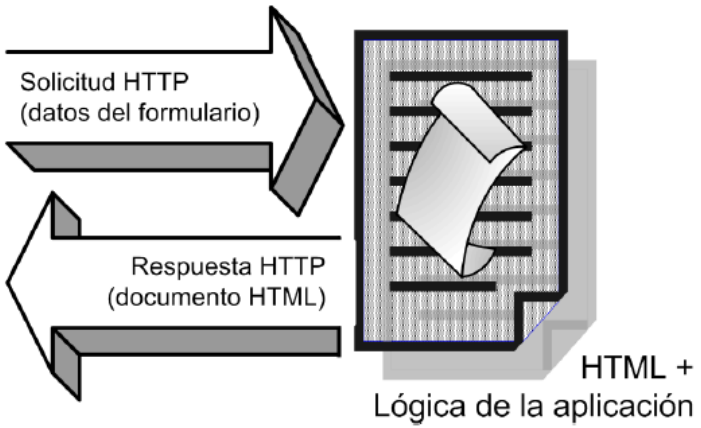
- Algunas de las herramientas y tecnologías que se pueden utilizar para construir aplicaciones web en el servidor son las siguientes:
  - Aplicaciones web compiladas: CGI (C y Perl)
  - Servlets (Java)
  - Aplicaciones web interpretadas: CGI scripts y Scripting languages (Perl, PHP, ColdFusion, Groovy)

ASP

- Es la tecnología de Microsoft que permite desarrollar aplicaciones web que ejecuten en el servidor HTTP de Microsoft, el Internet Information Server (IIS).
- Consiste en intercalar macros o fragmentos de código dentro de los documentos HTML que sirven para crear las interfaces de usuario de las aplicaciones web

ASP

- Una página ASP no es más que un fichero HTML con extensión .asp (.aspx en el caso de ASP.NET) al que le añadimos algo de código.



Funcionamiento de las páginas de servidor: Una página ASP/JSP contiene HTML estático intercalado con scripts que se encargan de generar HTML de forma dinámica.

ASP

- Para desarrollar páginas ASP con comodidad, el programador dispone de una serie de objetos predefinidos que simplifican su trabajo ocultando los detalles de la comunicación del navegador web del cliente con el servidor HTTP.

ASP

Objeto	Encapsula
Request	La solicitud HTTP recibida
Response	Las respuesta HTTP generada
Server	El estado del servidor
Application	El estado de la aplicación web
Session	La sesión de usuario

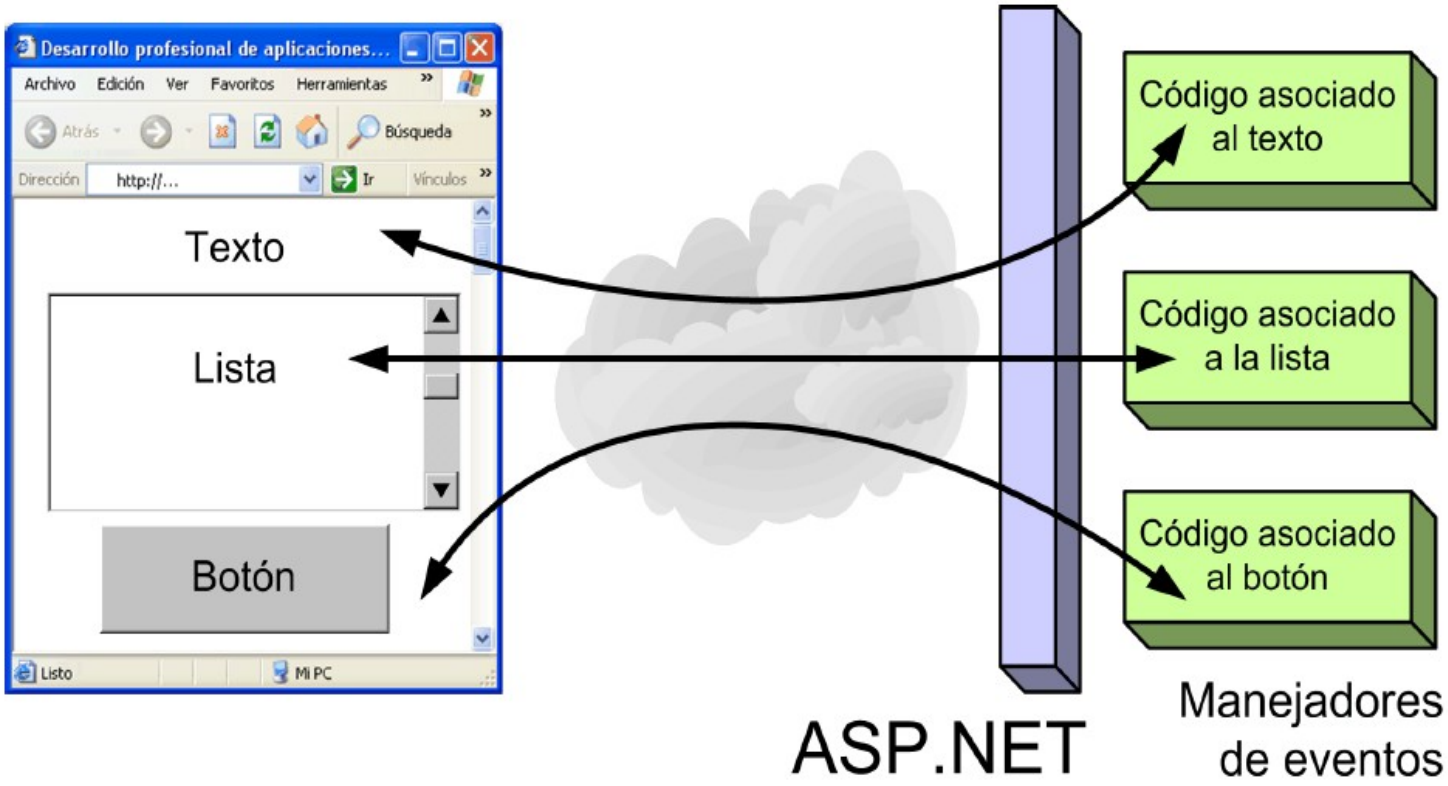
Ejemplo ASP .NET

```
<%@ Page language="c#" %>
<html><head>
        <title>Hora.aspx</title>
    </head>
<script runat="server">
public void Button_Click (object sender, System.EventArgs e)
    {
        LabelHora.Text = "La hora actual es " + DateTime.Now;
    }
</script>
<body><form method="post" runat="server">
        <asp:Button onclick="Button_Click" runat="server"
        Text="Pulse el botón para consultar la hora"/>
    <p>  <asp:Label id=LabelHora runat="server" />
    </form>
</body>
</html>
```

Páginas ASP .NET

- La programación en ASP.NET está basada en el uso de controles y eventos
- Las páginas ASP.NET, en vez de aceptar datos de entrada y generar su salida en HTML como sucede en ASP, implementan su funcionalidad en fragmentos de código que se ejecutan como respuesta a eventos asociados a los controles de la interfaz con los que puede interactuar el usuario.
- Requiere menos código y permite crear aplicaciones más modulares, legibles y mantenibles.

Páginas ASP .NET



Eventos en ASP.NET: La respuesta de la aplicación web se obtiene como resultado de ejecutar los manejadores de eventos asociados a los controles incluidos en la interfaz de usuario.

Páginas ASP .NET

- ASP.NET nos permite utilizar dos estilos bien diferenciados para la confección de páginas ASP.NET:
  - El primero de ellos consiste en incluir tanto los controles como el código en un único fichero .aspx, tal y como hicimos en el ejemplo anterior. Aunque esta forma nos impide aprovechar al máximo las ventajas de ASP.NET, por tenerlo todo en un fichero único.
  - ASP.NET también nos permite mantener los controles de nuestra interfaz en un fichero .aspx y dejar todo el código en un lugar aparte [*code-behind page*].

Ejemplo ASP .NET

- Veamos como quedaría el ejemplo anterior si hiciéramos dicha separación.
- En el fichero Ejemplo1.aspx, tendríamos el siguiente código:

Ejemplo ASP .NET

```
<%@ Page language="c#" Codebehind="HoraWebForm.aspx.cs"
        Inherits="HoraWeb.WebForm" %>
<html>
    <head>
        <title>Hora.aspx</title>
    </head>
    <body>
        <form method="post" runat="server">
            <asp:Button id="ButtonHora" runat="server"
                Text="Pulse el botón para consultar la hora" />
        <p><asp:Label id="LabelHora" runat="server" />
        </form>
    </body>
</html>
```

Ejemplo ASP .NET

- Una vez que tenemos el diseño de nuestro formulario, sólo nos falta implementar su comportamiento dinámico en un fichero aparte. Este fichero de código lo escribiremos como cualquier otro fichero de código en C#, creando para nuestra página una clase que herede de System.Web.UI.Page.

Ejemplo ASP .NET

```
namespace HoraWeb
{
    public class WebForm : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Button ButtonHora;
        protected System.Web.UI.WebControls.Label LabelHora;
        override protected void OnInit(EventArgs e)
        {
            this.ButtonHora.Click += new
            System.EventHandler(this.ButtonHora_Click);
            base.OnInit(e);
        }
        private void ButtonHora_Click(object sender, System.EventArgs e)
        {
            LabelHora.Text = "La hora actual es "+DateTime.Now;
        }}}
```

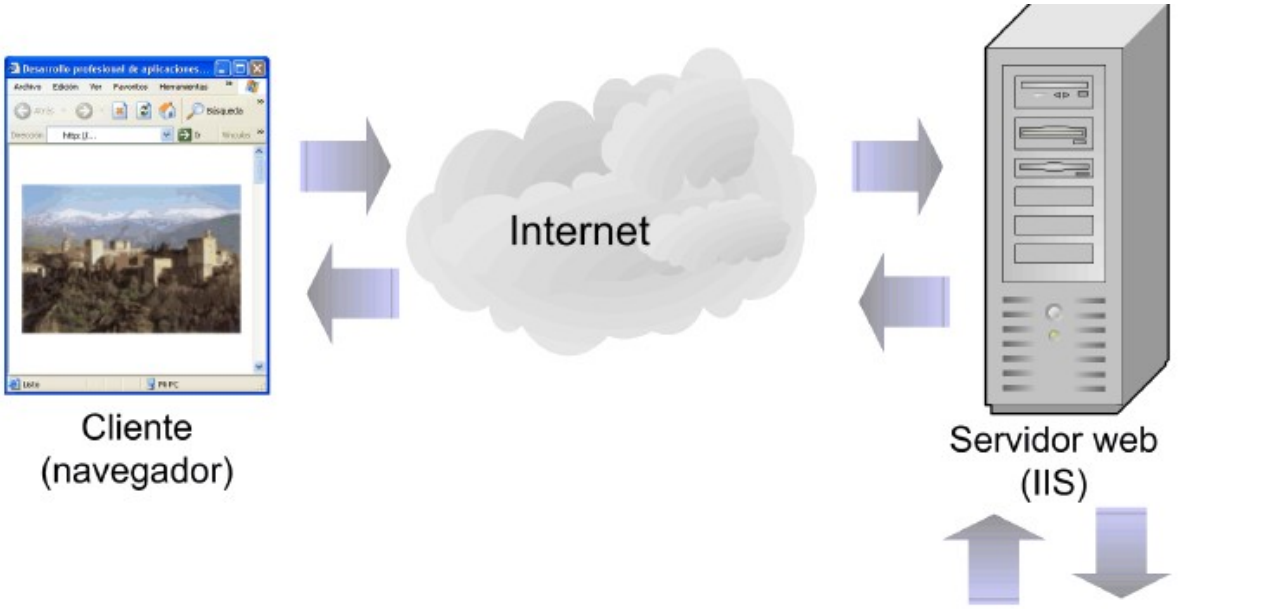
Formularios Web

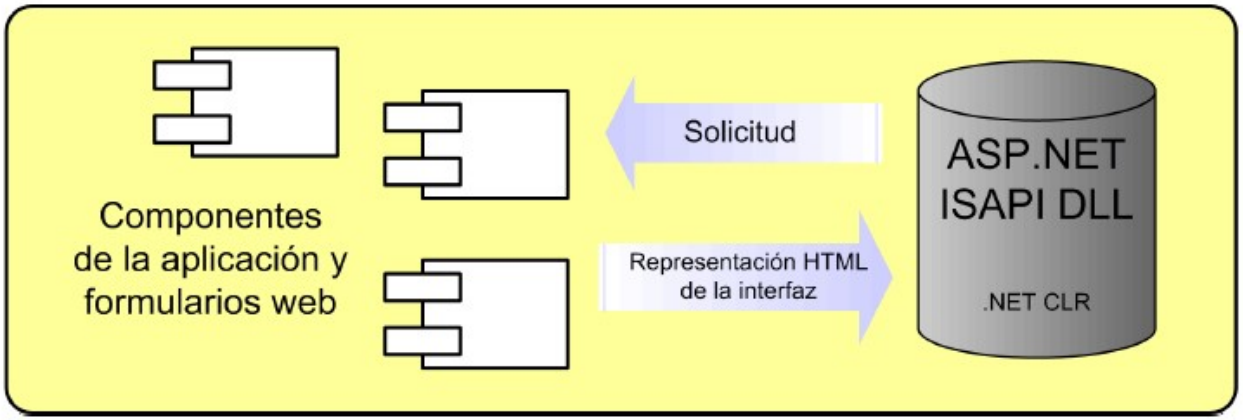
- Ejecución de páginas ASP .NET
  - Los servidores HTTP pueden configurarse de tal forma que las peticiones recibidas se traten de diferentes formas en función del tipo de recurso solicitado.
  - En el caso de las páginas ASP convencionales, cuando el usuario intenta acceder a un fichero con extensión .asp, el Internet Information Server delega en la biblioteca asp.dll, que se encarga de interpretar la página ASP.
  - Cuando se utiliza ASP.NET, el IIS se configura de tal forma que las solicitudes recibidas relativas a ficheros con extensión .aspx son enviadas a la biblioteca aspnet_isapi.dll.

Formularios Web

- Ejecución (cont.)
  - La biblioteca encargada de la ejecución de las páginas ASP.NET (aspnet_isapi.dll) encapsula el CLR [*Common Language Runtime*] de la plataforma .NET.
  - Así, podemos utilizar todos los recursos de la plataforma .NET en el desarrollo de aplicaciones web.
  - La DLL mencionada creará las instancias que sean necesarias de las clases .NET para atender las solicitudes recibidas en el servidor web.

Formularios Web





Creación de páginas ASP. NET

- La biblioteca de clases .NET incluye un conjunto de clases que nos serán de utilidad en la creación de las páginas ASP.NET.
- Entre dichas clases se encuentra una amplia gama de controles que podremos utilizar en la construcción de interfaces web para nuestras aplicaciones, controles tales como botones, cajas de texto, listas o tablas.
- Una aplicación web, generalmente, estará formada por varios formularios web. Cada uno de esos formularios lo implementaremos como una páginas ASP.NET.

Creación de páginas ASP .NET

- Para mantener independientes la interfaz de usuario y la lógica asociada a la aplicación, la implementación de las páginas ASP.NET la dividiremos en dos ficheros:
  - Un fichero con extensión .aspx especificaremos el aspecto de nuestra interfaz, utilizando tanto etiquetas HTML estándar como etiquetas específicas para hacer referencia a los controles ASP.NET que deseemos incluir en nuestra página.
  - En un segundo fichero, que será un fichero de código con extensión .cs si utilizamos en lenguaje C#, implementaremos la lógica de la aplicación.

Ejemplo fichero .aspx

- El siguiente ejemplo muestra el aspecto que tendrá nuestro fichero .aspx:

```
<% @Page Language="C#" Inherits="TodayPage" Src="Today.cs" %>

<html>
<body>
   <h1 align="center">
      Hoy es <% OutputDay(); %>
   </h1>
</body>
</html>
```

El fichero anterior incluye todo lo necesario para generar dinámicamente una página web en la que incluiremos la fecha actual, que aparecerá donde está el fragmento de código delimitado por <% y %>

Ejemplo fichero .cs

- El aspecto de este fichero de código, con extensión .cs, será el siguiente:

```
using System;
using System.Web.UI;

public class TodayPage:Page
{
  protected void OutputDay()
  {
    Response.Write(DateTime.Now.ToString("D"));
  }
}
```

Este fichero define una clase (TodayPage) que hereda de la clase System.Web.UI.Page. En nuestra clase hemos incluido un método que se encargará de generar un mensaje en el que se muestre la fecha actual

Uso de controles ASP .NET

- Vamos a utilizar controles definidos en la biblioteca de clases .NET.
- Así, las páginas ASP.NET se convierten en meras colecciones de controles.
- Desde el punto de vista del programador, cada control será un objeto miembro de la clase que representa la página .
- Cuando deseemos asociar a nuestra página un comportamiento dinámico, tenemos que asociar a los distintos controles los manejadores de eventos que se encargarán de implementar la funcionalidad de la página, exactamente igual que cuando se crea una interfaz gráfica para Windows utilizando un entorno de programación visual.

Uso de controles ASP .NET

- Los controles web constituyen una capa intermedia entre el código de la aplicación y la interfaz de usuario.
- Entre las ventajas que proporciona este hecho, destaca la compatibilidad automática de las aplicaciones web con distintos tipos de navegadores.
- Dentro de las páginas ASP.NET, los controles se indican en el fichero .aspx utilizando etiquetas de la forma <asp:... />.

Uso de controles ASP .NET

- Si estamos utilizando Visual Studio .NET como entorno de desarrollo, creamos un nuevo proyecto de tipo *Aplicación Web ASP.NET.*
- Al aparecernos un formulario web vacío, modificamos su propiedad pageLayout para utilizar el estilo FlowLayout típico de las páginas web, en las que los controles no se colocan en coordenadas fijas

Uso de controles ASP .NET

- Añadimos una etiqueta [Label] que obtenemos del cajón *Web Forms* del cuadro de herramientas.
- Una vez que tenemos el control en nuestro formulario web, lo normal es que modifiquemos su identificador (propiedad (ID)) y el texto que muestra en el formulario (propiedad Text).
- Como resultado de este proceso obtenemos el siguiente fichero .aspx, en el que aparece nuestro control como una etiqueta asp:Label:

Ejemplo formulario web con Visual Studio

```
<%@ Page language="c#"   Codebehind="WebControlExample.aspx.cs"
AutoEventWireup="false"   Inherits="WebControlExample.WebForm" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
      <HEAD>
            <title>WebForm</title>
            <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
            <meta name="CODE_LANGUAGE" Content="C#">
      </HEAD>
      <body>
            <form id="Form1" method="post" runat="server">
                  <p align="center">
                        <asp:Label id="LabelFecha" runat="server">Hoy es...</asp:Label>
                  </p>
            </form>

      </body>

</HTML>
```

Uso de controles ASP .NET

- **IMPORTANTE**
  - Todos los controles en una página ASP.NET deben estar dentro de una etiqueta <form> con el atributo runat="server".
  - Además, ASP.NET requiere que todos los elementos HTML estén correctamente anidados y cerrados (como sucede en XML).

Ejemplo de formulario web

- Acto seguido, implementamos la lógica de la aplicación como respuesta al evento dentro del fichero de código C# asociado a la página ASP.NET.
- En el ejemplo que nos ocupa sólo tenemos que escribir una línea de código dentro del método creado automáticamente por el Visual Studio .NET para gestionar el evento Page_Load:

Formularios web en ASP .NET

- Vamos a pasar a analizar los distintos tipos de controles que podemos incluir en nuestras páginas web. En la plataforma .NET, existen tres tipos de controles predefinidos que se pueden utilizar en las páginas ASP.NET:
  - Los controles HTML representan etiquetas HTML tradicionales y funcionan de forma similar a los objetos utilizados en JavaScript para manipular dinámicamente el contenido de una página web.
  - Los controles web son los controles asociados a las etiquetas específicas de ASP.NET y facilitan el desarrollo de interfaces web utilizando un entorno de programación visual como Visual Studio .NET.
  - Por último, existe una serie de controles de validación que se emplean para validar las entradas introducidas por el usuario

Controles HTML

- Por defecto, las etiquetas HTML incluidas en una página ASP.NET se tratan como texto en el servidor web y se envían tal cual al cliente.
- Para que las etiquetas HTML sean programables en el servidor, sólo tenemos que añadirles el atributo runat="server".

Controles HTML

- Una vez que hemos marcado el enlace con el atributo runat="server", podemos modificar sus propiedades accediendo a él a través del identificador que le hayamos asociado.
- En ASP.NET, los enlaces HTML de una página se representan mediante el control HtmlAnchor. Una de las propiedades de este control (HRef) indica la URL a la que apunta el enlace, por lo que sólo tenemos que establecer un valor adecuado para esta propiedad en el código asociado a alguno de los eventos de la página ASP.NET.
- El fichero de código resultante tendría el siguiente aspecto:

Controles HTML

```
public class HTMLControl : System.Web.UI.Page
{
    protected System.Web.UI.HtmlControls.HtmlAnchor enlace;
    private void Page_Load(object sender, System.EventArgs e)
    {
        enlace.HRef = "http://csharp.ikor.org/";
    }
    override protected void OnInit(EventArgs e)
    {
        this.Load += new System.EventHandler(this.Page_Load);
        base.OnInit(e);}}
```

Controles Web

- La principal aportación de ASP.NET a la creación de interfaces web es la inclusión de controles específicos que aíslan al programador del HTML generado para presentar el formulario al usuario de la aplicación.

Controles Web

Control	Descripción
AdRotator	Muestra una secuencia de imágenes (a modo de banner)
Button	Botón estándar
Calendar	Calendario mensual
CheckBox	Caja de comprobación (como en los formularios Windows)
CheckBoxList	Grupo de cajas de comprobación
DataGrid	Rejilla de datos
DataList	Muestra una lista utilizando plantillas (*templates)*
DropDownList	Lista desplegable
HyperLink	Enlace
Image	Imagen

Controles Web

Control	Descripción
ImageButton	Botón dibujado con una imagen
Label	Etiqueta de texto estático
LinkButton	Botón con forma de enlace
ListBox	Lista (como en los formularios Windows)
Literal	Texto estático (similar a Label)
Panel	Contenedor en el que se pueden colocar otros controles
PlaceHolder	Reserva espacio para controles añadidos dinámicamente
RadioButton	Botón de radio (como en los formularios Windows)
RadioButtonList	Grupo de botones de radio

Controles Web

Control	Descripción
Repeater	Permite mostrar listas de controles
Table	Tabla
TextBox	Caja de edición
Xml	Muestra un fichero XML o el resultado de una transformación XSL

Controles de Validación

- Estos controles se enlazan a controles ASP.NET de los descritos en el apartado anterior para validar las entradas de un formulario web.
- Cuando el usuario rellena los datos de un formulario y alguno de los datos introducidos no verifica la restricción impuesta por el control de validación, este control se encarga de mostrarle un mensaje de error al usuario.

Controles de Validación

- Para que nuestro formulario web utilice el control de validación, sólo tenemos que establecer la propiedad ControlToValidate de este control para que haga referencia al TextBox que usamos como entrada.
- Mediante la propiedad ErrorMessage especificamos el mensaje de error que se mostrará (en rojo) cuando el valor introducido por el usuario no cumpla la condición impuesta por el control de validación.

Controles de Validación

- Por cuestiones de seguridad, los controles de validación siempre validan los datos de entrada en el servidor, con el objetivo de que un cliente malintencionado no pueda saltarse algunas comprobaciones acerca de la validez de los datos.

Controles de Validación

Control de validación	Descripción
`CompareValidator`	Compara el valor de una entrada con el de otra o un valor fijo.
`CustomValidator`	Permite implementar un método cualquiera que maneje la validación del valor introducido.
`RangeValidator`	Comprueba que la entrada esté entre dos valores dados.
`RegularExpressionValidator`	Valida el valor de acuerdo a un patrón establecido como una expresión regular.
`RequiredFieldValidator`	Hace que un valor de entrada sea obligatorio.

Creación de un servicio web

- Los servicios Web son componentes de un servidor web a los que puede llamar una aplicación cliente realizando solicitudes HTTP a través de Internet.
- ASP.NET permite crear servicios Web personalizados o utilizar servicios de aplicación integrados y llamar a estos servicios desde cualquier aplicación cliente.

Creación de un servicio web

- Si crea un servicio Web con código administrado basado en ASP.NET y .NET Framework, no es necesario escribir *código de infraestructura* para administrar detalles como los protocolos de comunicaciones o transportes de mensajes.
- Es más, si se crean servicios Web que usen el marco de páginas ASP.NET, los servicios pueden aprovechar muchas de las características de .NET Framework, como autenticación, almacenamiento en caché y administración de estados.
- En el modelo de aplicación ASP.NET, las páginas web usan la extensión .aspx. Para diferenciar los servicios Web de las páginas ASP.NET habituales, los servicios utilizan la extensión .asmx.

Creación de un servicio web

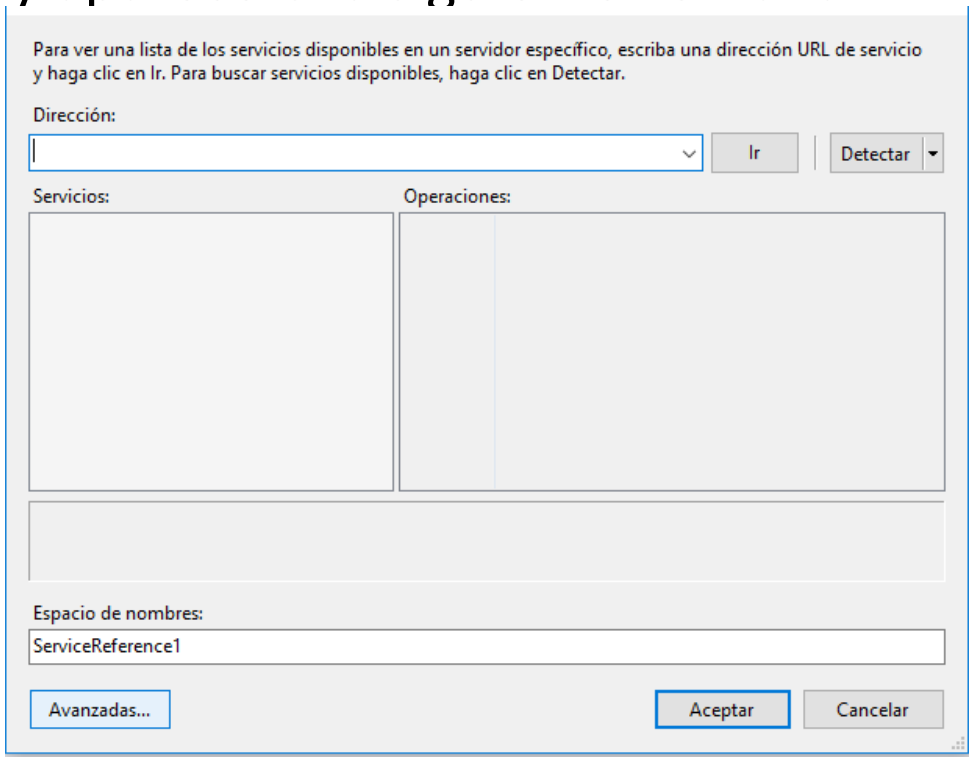
- Los servicios Web pueden ser aplicaciones independientes o subcomponentes de una aplicación web más grande.
- A continuación vamos a ver paso a paso la creación de una aplicación sencilla que incluye un servicio web.

Creación de un servicio web

- Reutilizaremos la aplicación que mostraba la hora en el label al darle al botón.
  - Tendremos el WebForm PruebaHora.aspx, y su correspondiente fichero de código asociado PruebaHora.aspx.cs.

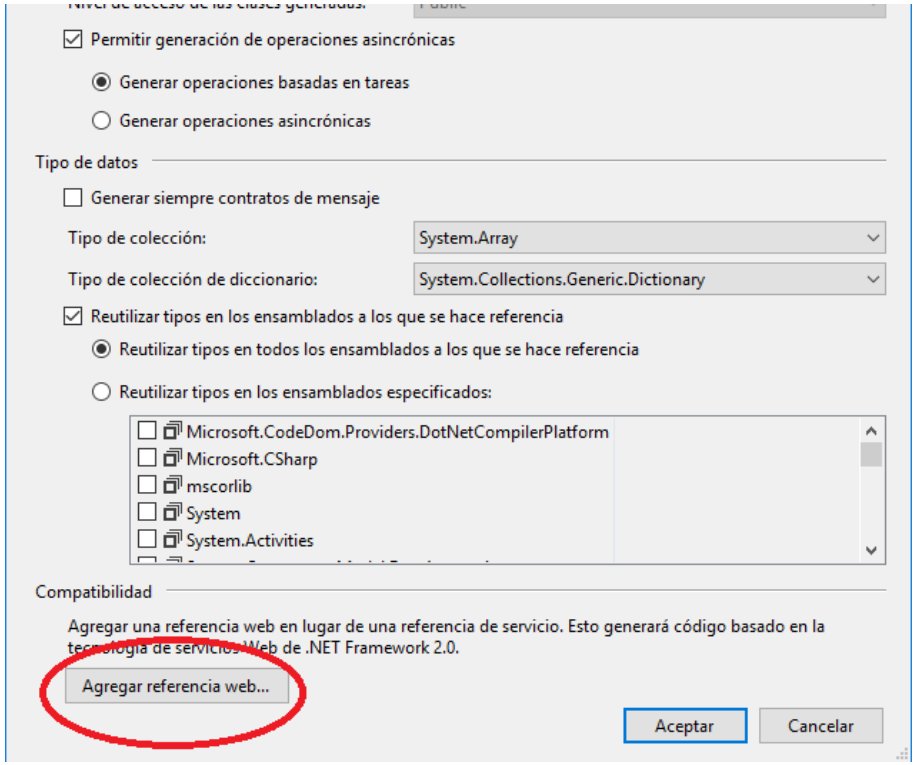
Creación de un servicio web

- Ahora pasamos a enlazar el cliente con un servicio.
  - Pulsamos en el menú de arriba, en la opción **Sitio Web > Agregar referencia de servicio,** y aparecerá la siguiente ventana:



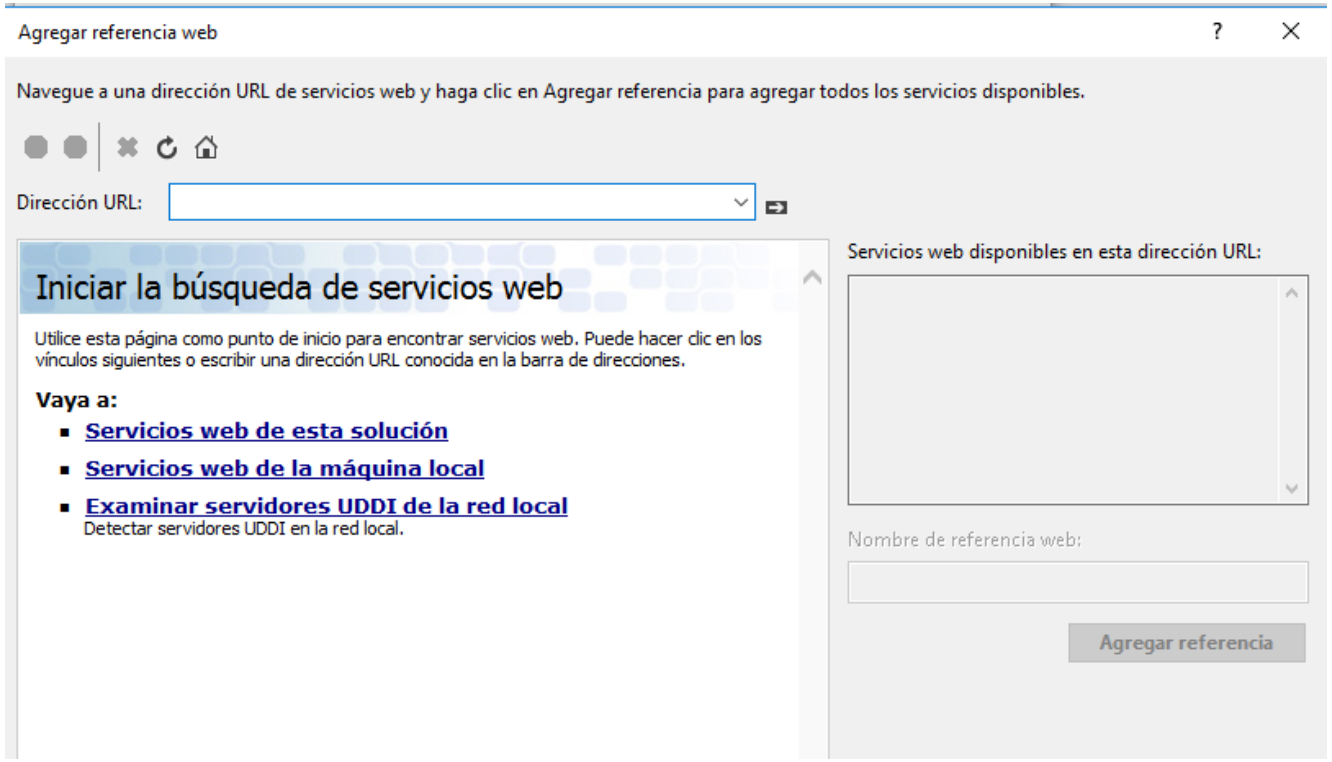
Creación de un servicio web

- Pulsamos en la opción de Avanzadas, y aparecerá la siguiente ventana:



Creación de un servicio web

- Seleccionamos **Agregar referencia web**, y se mostrará la siguiente ventana:



Creación de un servicio web

- Hay tres opciones:
  - Servicios Web de esta solución.
  - Servicios Web del equipo local.
  - Examinar servidores UDDI de la red local.
- De estas tres opciones, inicialmente utilizaremos la primera, es decir utilizar un servicio web de esta solución.
- Actualmente no tenemos ningún servicio web en nuestra solución, por eso pasaremos a añadir dicho servicio.

Creación de un servicio web

- Agregamos a nuestro proyecto un nuevo servicio, seleccionamos la raíz del proyecto, le damos a botón derecho, y seleccionamos agregar nuevo elemento, que en este caso, será **servicio web.**
- Una vez hagamos esto, en nuestro explorador de soluciones, aparecerá, una carpeta **App_Code,** que contendrá el fichero cs, y además se añade el fichero .asmx.

Creación de un servicio web

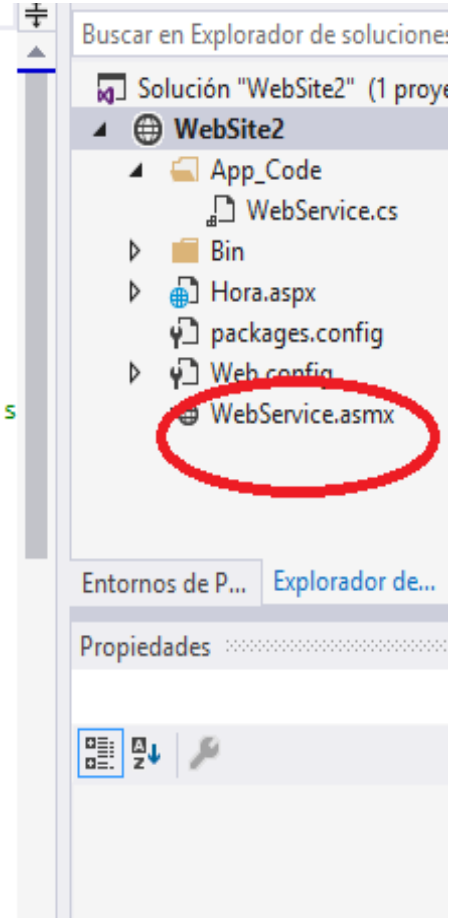
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

/// <summary>
/// Descripción breve de WebService
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// Para permitir que se llame a este servicio web desde un script, usando ASP.NET AJAX, quite la marca de comentario de la línea s
// [System.Web.Script.Services.ScriptService]
public class WebService : System.Web.Services.WebService
{

    public WebService()
    {

        //Elimine la marca de comentario de la línea siguiente si utiliza los componentes diseñados
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld()
    {
```



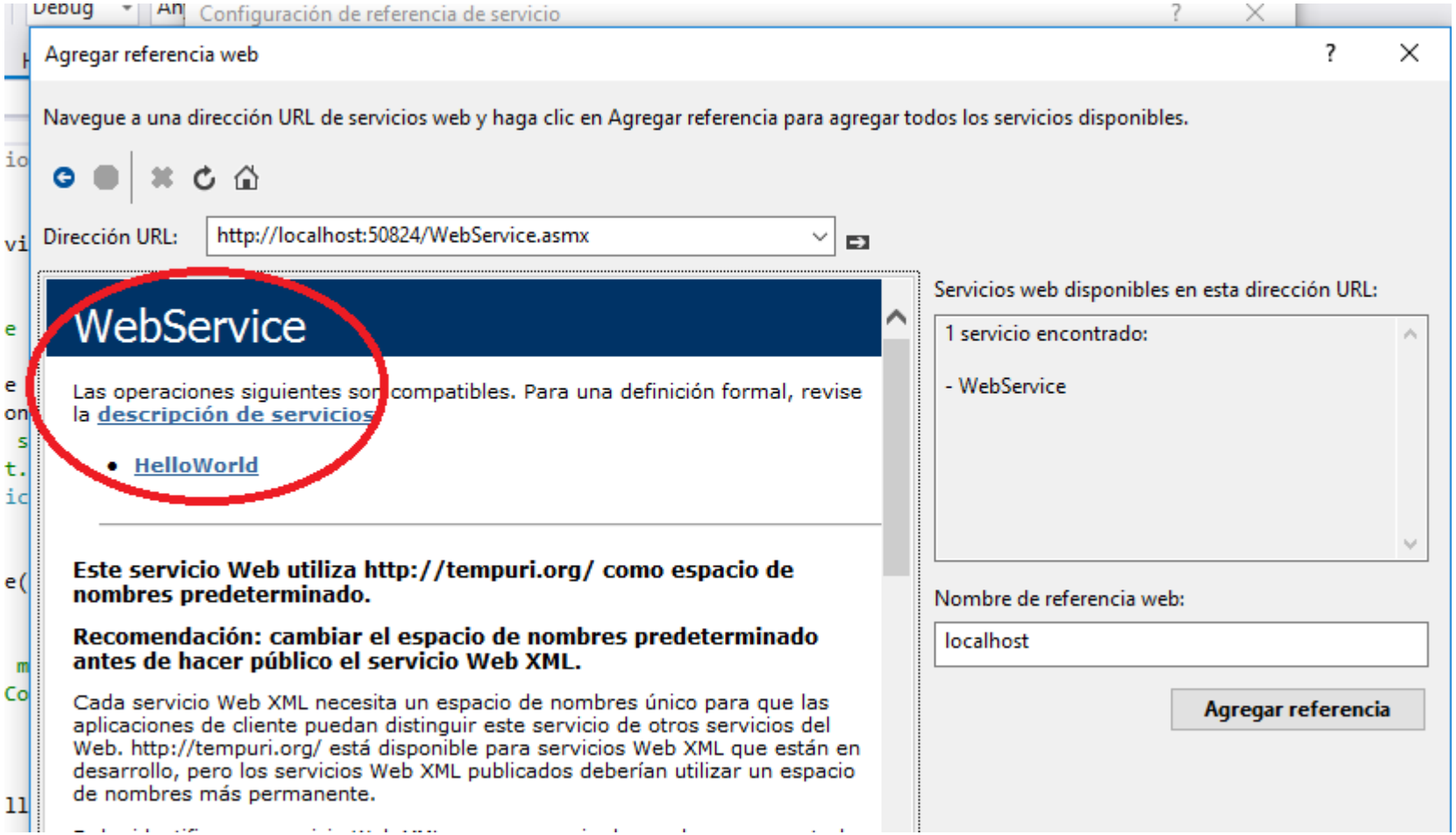
Creación de un servicio web

- El servicio web, no deja de ser una clase que hereda de otra, concretamente de System.Web.Services.WebService.
- Esa clase tiene un **WebMethod**, que lo único que hace es mostrar un mensaje de "Hola a todos". Si quisiéramos darle más funcionalidad a nuestro servicio, podríamos añadir nuevos WebMethod, pero **ATENCIÓN**, es muy importante mantener la sintaxis a la hora de crear nuevos WebMethods, puesto que si no podemos provocar errores.

Creación de un servicio web

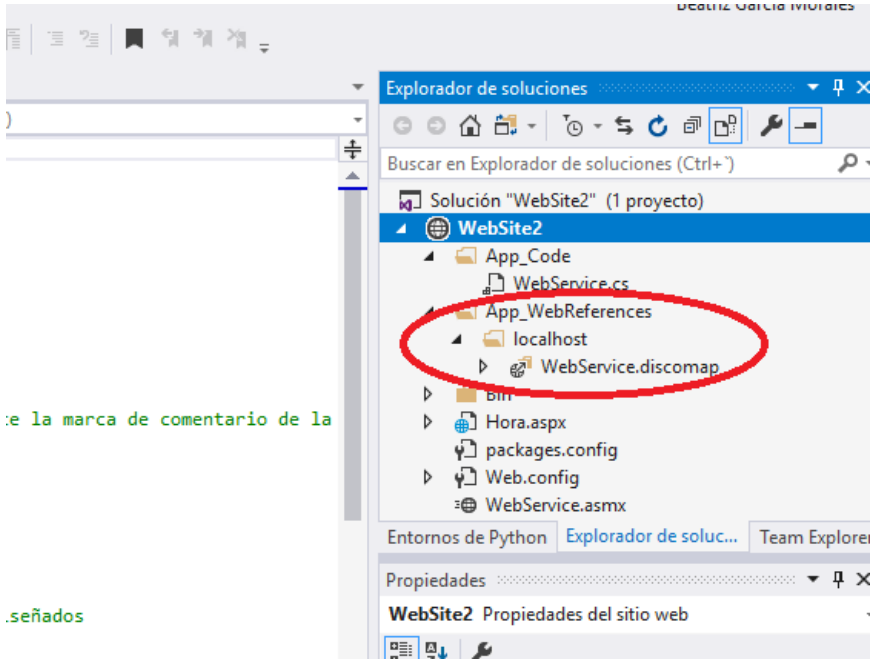
- Cuando ahora volvamos a realizar el paso de agregar referencia web, y seleccionemos la opción Servicio de esta solución, ya sí aparecerá el servicio que hemos añadido a nuestra solución
- Aparecerá ahora esta ventana:

Creación de un servicio web



Creación de un servicio web

- Al darle a botón de Agregar referencia, en nuestro explorador de soluciones, aparece una nueva carpeta **App_WebReferences**, con diferentes elementos, lo que nos indica que la referencia se ha añadido correctamente.



Creación de un servicio web

- Ahora ya podemos invocar desde el cliente a los webmethods. En nuestro caso al pulsar el botón Hora, invocaremos el método MuestraHora, que quedaría de la siguiente forma:

```
[WebMethod]
    public string MuestraHora()
    {
        string mensaje;
        mensaje = "La hora actual es:" +DateTime.Now.ToShortTimeString();
        return mensaje;
    }
```

Creación de un servicio web

- En el webForm, en el evento click del botón, pondríamos el siguiente código:

```
protected void Button1_Click(object sender, EventArgs e)
    {
        WebService s = new WebService();
        Label1.Text = s.MuestraHora();
    }
```

Creación de un servicio web

- Al ejecutar la aplicación, el resultado será el mismo, pero tendremos un servicio que se encarga de realizar todas las operaciones que solicita el usuario desde la interfaz a través del WebForm.