



1.- QUÉ ES C#	2
C# intenta ser el lenguaje base para escribir aplicaciones .NET	2
2.- DEFINICIONES BÁSICAS.....	2
NameSpace	2
Utilizar Namespace en Aplicaciones Cliente	2
Using.....	2
Clases.....	3
Eventos.....	3
Constructor	4
3.- PRIMERA CLASE: HOLA MUNDO.....	4
3.1 Hola mundo usando codefile (usando Page_Load)	6
3.2 Hola mundo usando usado en flujo de ASPX.....	7
3.3 Agregar una clase (DLL) a un sitio web.....	8
4.- MASTER.PAGE.....	10
Master Pages y Content Pages.....	10
“URL Rebasing” en Master Pages.....	12
Anidando Master Pages	13
5.- CONTROLES EN UNA CONTENT.PAGE	14
Vamos a distinguir las partes de la ventana:	14
Empezando con la aplicación.....	15
Escribir el código.....	16
6.- EJERCICIO	18



1.- QUÉ ES C#

C# o C Sharp es un lenguaje de programación que está incluido en la Plataforma .NET y corre en el Lenguaje Común en Tiempo de Ejecución (CLR, Common Language Runtime). El primer lenguaje en importancia para el CLR es C#, mucho de lo que soporta la Plataforma .NET está escrito en C#.

C# intenta ser el lenguaje base para escribir aplicaciones .NET

C# deriva de C y C++, es moderno, simple y **enteramente orientado a objetos**, simplifica y moderniza a C++ en las áreas de clases, namespaces, sobrecarga de métodos y manejo de excepciones. Se eliminó la complejidad de C++ para hacerlo más fácil de utilizar y menos propenso a errores.

C# es "case sensitive", es decir, que distingue mayúsculas de minúsculas. HolaMundo es diferente a holamundo.

2.- DEFINICIONES BÁSICAS.

NameSpace

Utilizados para organizar las clases y otros tipos en una estructura jerárquica. El propósito del uso de namespace hacen las clases fáciles de usar y prevenir colisiones con las clases escritas por otros programadores.

Un namespace contiene tipos que pueden ser utilizados en la construcción de programas: clases, estructuras, enumeraciones, delegados e interfaces, por ejemplo para poder escribir a la consola se utiliza el namespace System.

En este curso no se crearán nameSpace, pero si se usarán cuando se importen clases (librerías) de otros programadores. El nameSpace de todos los componentes de la universidad es "ua".

Los nombres de espacio son definidos utilizando la sentencia: namespace

Utilizar Namespace en Aplicaciones Cliente

Al desarrollar componentes utilizando namespace la aplicación cliente debe importarlo:
using ua;

Using

Las colisiones entre tipos o nombres de espacio que tienen el mismo nombre se pueden resolver utilizando una variante de la cláusula using que permite definir un alias para la clase:



```
using Alias = System.Web.HttpContext;
class NombreClase{
    public static void Main()
    {
        Alias.Current.Response.Write ("Alias de una clase");
    }
}
```

Clases

Una Clase es una plantilla para un objeto.

Una Clase define las operaciones que un objeto puede realizar y define un valor que mantiene el estado del objeto, los componentes principales de una clase son:

métodos, eventos y propiedades.

Una instancia de una clase es un objeto, se accede a la funcionalidad de un objeto invocando sus métodos y accediendo a sus propiedades, eventos y campos.

Existe una palabra reservada llamada **this** que sirve para hacer referencia a la clase actual en el ámbito en el cual es utilizada. Cuando se hace referencia a una variable de instancia que tiene el mismo nombre de un parámetro se debe utilizar `this.name`.

Al crear y manipular objetos no es necesario administrar la memoria que estos ocupan -ya que existe un mecanismo que se encarga de esto llamado garbage collector (recolector de basura), pero es una buena práctica no olvidar liberar los recursos.

Según la guía de estilo del CPD de la Universidad de Alicante se requiere que el nombre de la clase siga una notación Pascal (ej: HolaMundo, Taxi).

Para indentar se usa TAB y no los espacios.

Las llaves deben estar en el mismo nivel que el código fuera de las llaves.

Se debe usar una línea en blanco para separar los grupos lógicos de código.

```
public class Taxi
{
    public bool isInitialized;
    public InicializaTaxi()
    {
        isInitialized = true;
    }
}
```

Eventos

Un método es un conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre. A estas instrucciones se les denomina cuerpo del método, y a su ejecución a través de su nombre se le denomina llamada al método.



La ejecución de las instrucciones de un método puede producir como resultado un objeto de cualquier tipo. A este objeto se le llama valor de retorno del método y es completamente opcional, pudiéndose escribir métodos que no devuelvan ninguno.

La ejecución de las instrucciones de un método puede depender del valor de unas variables especiales denominadas **parámetros** del método, de manera que en función del valor que se dé a estas variables en cada llamada la ejecución del método se pueda realizar de una u otra forma y podrá producir uno u otro valor de retorno.

Al conjunto formado por el nombre de un método y el número y tipo de sus parámetros se le conoce como **signatura del método**. La signatura de un método es lo que verdaderamente lo identifica, de modo que es posible definir en un mismo tipo varios métodos con idéntico nombre siempre y cuando tengan distintos parámetros. Cuando esto ocurre se dice que el método que tiene ese nombre está **sobrecargado**.

Según la guía de estilo se requiere que el nombre del evento siga una notación Pascal (ej: EscribeHolaMundo, InicializaTaxi).

Constructor

Los constructores son métodos de clase que se ejecutan cuando se crea un objeto de un tipo determinado. Los constructores tienen el mismo nombre que la clase y, normalmente, inicializan los miembros de datos del nuevo objeto.

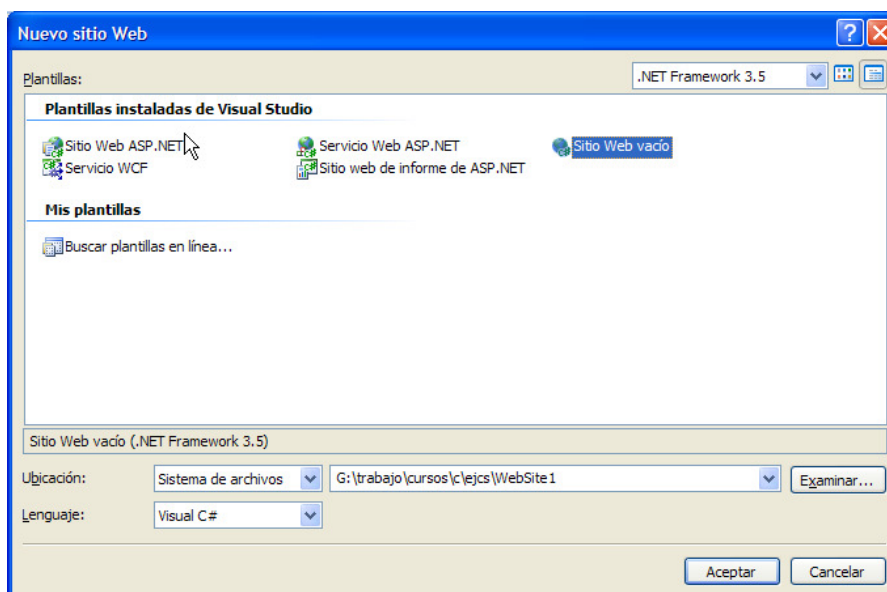
En el ejemplo siguiente, una clase denominada Taxi se define con un constructor simple. Esta clase crea instancias con el operador new. El operador new invoca el constructor Taxi inmediatamente después de asignar la memoria al nuevo objeto

```
HolaMundo InicioHolaMundo = new HolaMundo();
```

Según la guía de estilo se requiere que el nombre del constructor siga una notación Pascal (ej: InicioHolaMundo, InicioTaxi).

3.- PRIMERA CLASE: HOLA MUNDO.

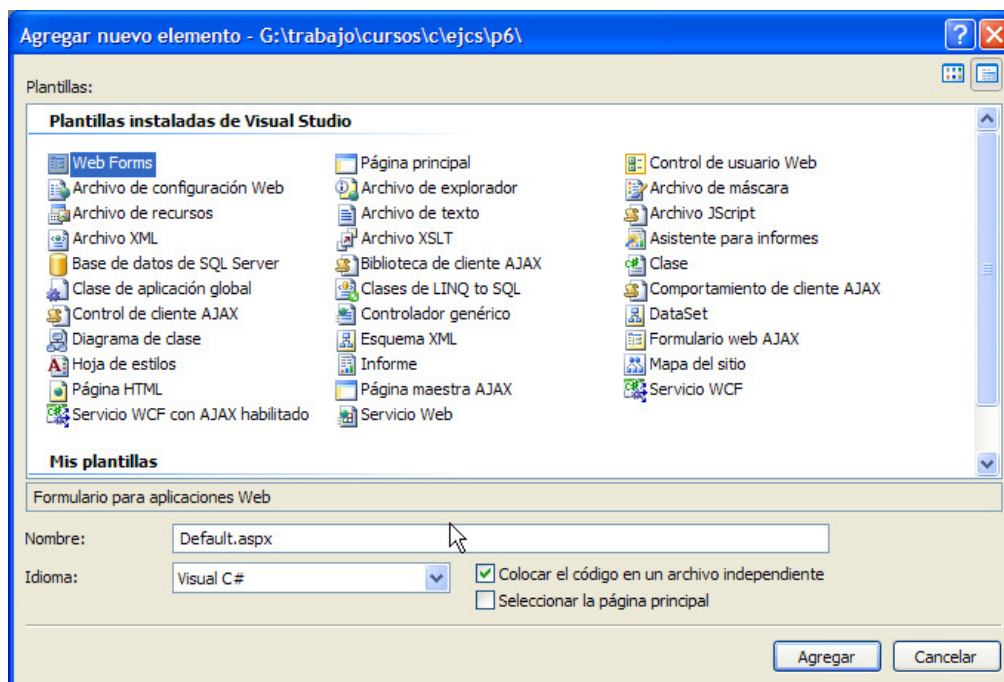
Archivo > Nuevo > Sitio Web >





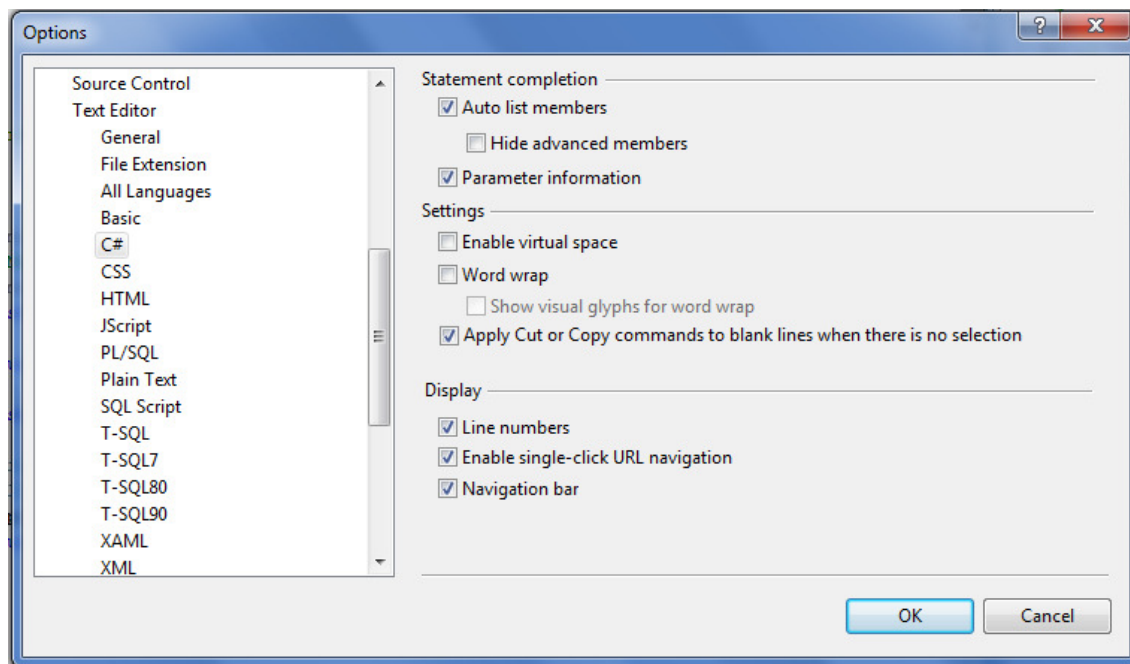
Poner ubicación y nombre.

Agregar nuevo elemento.

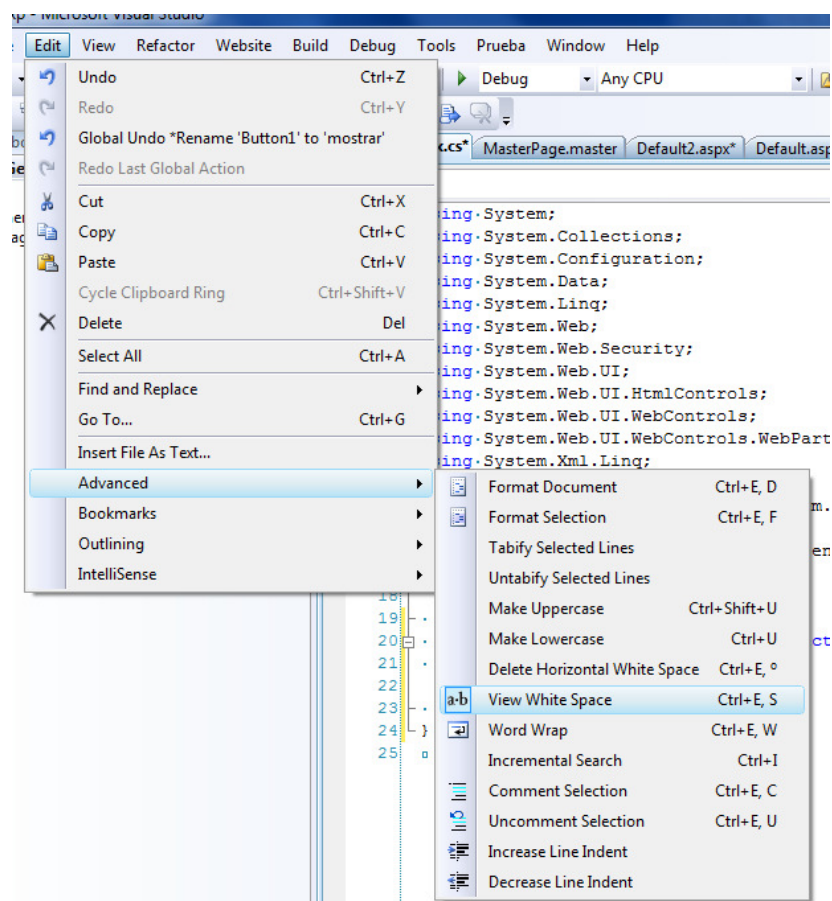


Para configurar el entorno que vemos podemos acudir a:

Herramientas >> Opciones >> Editor de texto >> c#



Otra opción complementaria es:



3.1 Hola mundo usando codefile (usando Page_Load)

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

Abrimos Default.aspx.cs

Copiamos

```
using System;
using System.Web;
public partial class _Default : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e)
    {
        HolaMundo InicioHolaMundo = new HolaMundo();
        HttpContext.Current.Response.Write(InicioHolaMundo.EscribeHolaMundo());
    }
}

///<summary>
/// Clase para escribir hola mundo
///</summary>
class HolaMundo {
    ///<summary>
    /// Método que sirve para escribir hola mundo (función)
    ///</summary>
    public string EscribeHolaMundo(){
        return "Hola mundo";
    }
}
```



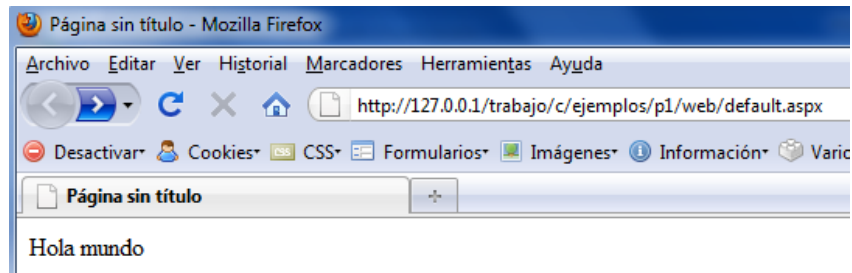
```
}
}
```

Observaciones:

El nombre de la clase sigue una notación Pascal (HolaMundo). El nombre del método también (EscribeHolaMundo) y el del constructor (InicioHolaMundo).

Resultado

Al cargar la página se escribirá "Hola Mundo"



3.2 Hola mundo usando usado en flujo de ASPX

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

Abrimos Default.aspx.cs y lo dejamos como sigue:

Copiamos

```
using System;
using System.Web;
public partial class _Default : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}

///<summary>
/// Clase para escribir hola mundo
///</summary>
class HolaMundo {
    ///<summary>
    /// Método que sirve para escribir hola mundo (función)
    ///</summary>
    public string EscribeHolaMundo(){
        return "Hola mundo";
    }
}
```

Abrimos Default.aspx y lo dejamos como sigue:

Copiamos



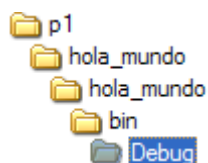
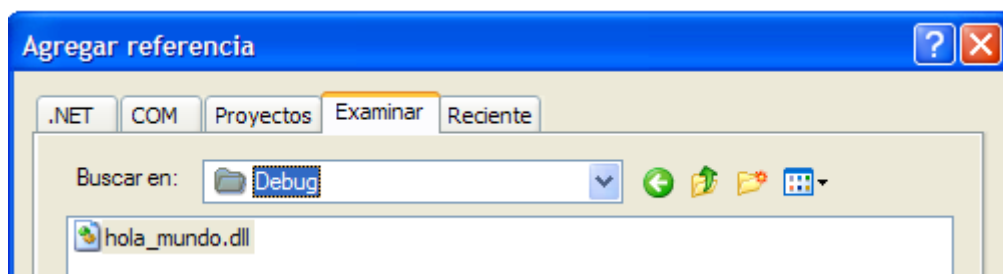
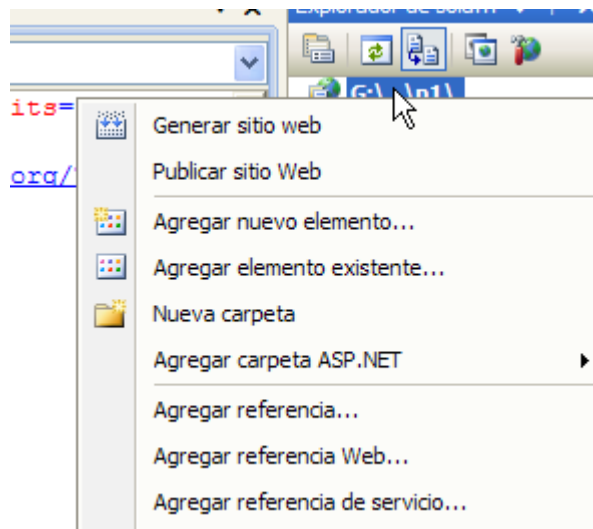
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default1.aspx.cs" Inherits="_Default" %>

<!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>Página sin título</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <%
        HolaMundo InicioHolaMundo = new HolaMundo();
        Response.Write(InicioHolaMundo.EscribeHolaMundo());
      %>
    </div>
  </form>
</body>
</html>
```

3.3 Agregar una clase (DLL) a un sitio web

Botón derecho >> Agregar referencia





```
using System;
using System.Web;
using ua;

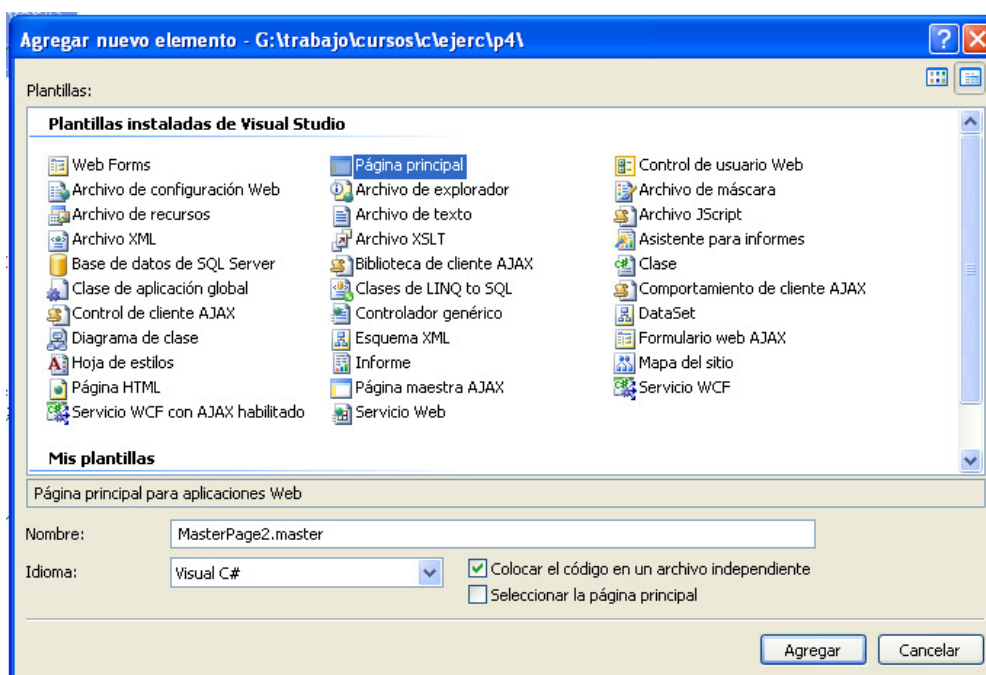
public partial class web_Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        HolaMundo IniciaHolaMundo = new HolaMundo();
        HttpContext.Current.Response.
            Write(IniciaHolaMundo.EscribeHolaMundo());
    }
}
```



4.- MASTER.PAGE

Una Master Page es una página que contiene marcas y controles que deben ser compartidas a través de páginas múltiples de nuestro sitio. Por ejemplo, si todas las páginas deben tener los mismos banners de cabecera y pie de página o el mismo menú de navegación, podemos definir esto en una Master Page una vez, de forma que todas las páginas asociadas a dicha Master Page heredarán estos elementos comunes.

La ventaja de definir la cabecera, el pie de página y la navegación en una Master Page es que estos elementos sólo tendrán que ser definidos una vez, en lugar de muchas veces y en código duplicado en las diferentes páginas del sitio.



Master Pages y Content Pages

La definición de una Master Page es como la de cualquier página. Las Master Pages pueden contener marcas, controles, código o cualquier combinación de estos elementos.

Sin embargo, una Master Page puede contener un tipo especial de control llamado **ContentPlaceHolder**. Un ContentPlaceHolder define una región de la representación de la master page que puede substituirse por el contenido de una página asociada a la maestra. Un ContentPlaceHolder también puede contener contenido por defecto, por si la página derivada no necesita sobrescribir este contenido. La sintaxis de un control ContentPlaceHolder es como sigue:



MASTER PAGE

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage2.master.cs"
Inherits="MasterPage2" %>
```

```
<!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>Página sin título</title>
```

```
<asp:ContentPlaceHolder id="head" runat="server">
```

```
</asp:ContentPlaceHolder>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div>
```

```
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
```

```
</asp:ContentPlaceHolder>
```

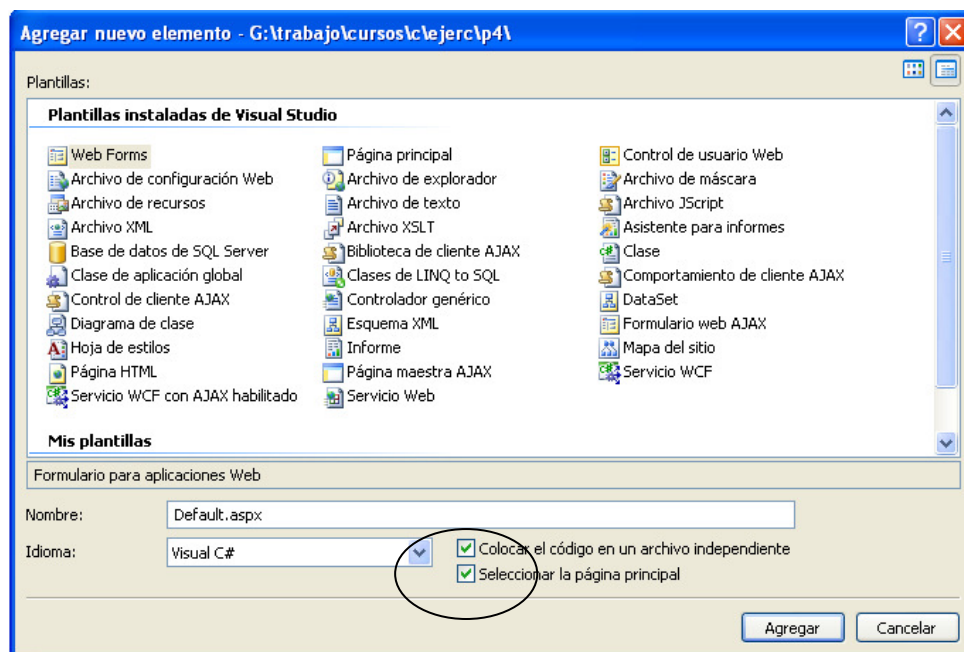
```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

Para diferenciar una Master Page de una página normal, la Master Page se guarda con una extensión .master. Una página puede derivar de una Master Page simplemente con definir un atributo MasterPageFile en su directiva Page, de la forma vista arriba. Una página que se asocia a una Master Page se llama Content Page (Página de Contenido)





CONTENT PAGE

```
<%@ Page MasterPageFile="Site.master" %>
```

```
<%@ Page Language="C#"
    MasterPageFile="~/MasterPage.master"
    AutoEventWireup="true"
    CodeFile="Default_.aspx.cs"
    Inherits="Default_"
    Title="Página sin título" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Una Content Page puede declarar controles Content que sobrescriban específicamente el contenido de las secciones marcadas en la Master Page.

Un control Content se asocia a un control ContentPlaceHolder particular a través de la propiedad ContentPlaceHolderID.

Una Content Page debe contener marcas y controles sólo dentro de los controles Content; no puede tener ningún contenido de alto nivel por sí misma. Puede, sin embargo, tener directivas o código del lado del servidor.

```
<%@ Page MasterPageFile="Site.master" %>
<asp:content id="Content1" contentplaceholderid="FlowerText" runat="server">
    Con sol, agua y cuidados las rosas aparecerán varias veces el la misma estación.
</asp:content>
<asp:content id="Content2" contentplaceholderid="FlowerPicture" runat="server">
    <asp:Image id="image1" imageurl="~/images/rose.jpg" runat="server"/>
</asp:content>
```

En este ejemplo muestra la relación entre las Master Pages y las Content Pages. La Master Page define, en este caso, dos regiones ContentPlaceHolder llamadas **FlowerPicture** y **FlowerText**, con algo de contenido por defecto para dichas regiones.

Las páginas individuales de contenido del sitio heredan el diseño común de sitio y el "ambiente" de la Master Page, pero sobrescriben el contenido por defecto de las regiones ContentPlaceHolder con su propio contenido. Observad que la página Default.aspx del sitio no define ningún control Content, de forma que heredo el contenido por defecto de la Master Page.

"URL Rebasing" en Master Pages

Una cosa a observar sobre los ejemplos anteriores es que hay varias páginas en una Master Page que hacen referencia a recursos URL, como imágenes, hojas de estilo o referencias a páginas, mediante la sintaxis de rutas relativas, por ejemplo:

```
<head>
<link rel="stylesheet" href="StyleSheet.css" type="text/css" />
</head>
...
```



```
<a href="daffodil.aspx">Daffodil</a>
```

```
...
```

```

```

Esto funciona cuando la Master Page y la Content Page se encuentran en el mismo directorio, pero cuando la Content Page se encuentra en una ubicación distinta, la ruta relativa no será correcta. Para solucionar este problema, podemos escoger una de las siguientes propuestas:

- Utilizar rutas URL absolutas en la Master Page, por ejemplo ``
- Utilizar URLs relativas o URLs relativas de aplicación en los controles de servidor en lugar de marcas estáticas, por ejemplo `<asp:Image ImageUrl="~/images/banner.gif" runat="server" />`

El siguiente ejemplo muestra esta técnica. Las Páginas de Contenido han sido movidas el subdirectorio "Pages" bajo el directorio que contiene la Página Maestra. La Página Maestra se ha actualizado para utilizar **controles de servidor** en lugar de HTML:

```
<head runat="server">
```

```
<link rel="stylesheet" href="StyleSheet.css" type="text/css" />
```

```
</head>
```

```
...
```

```
<a id="A1" href="pages/daffodil.aspx" runat="server">Daffodil</a>
```

```
...
```

```
<asp:Image ID="Image1" AlternateText="Water Lillies"
```

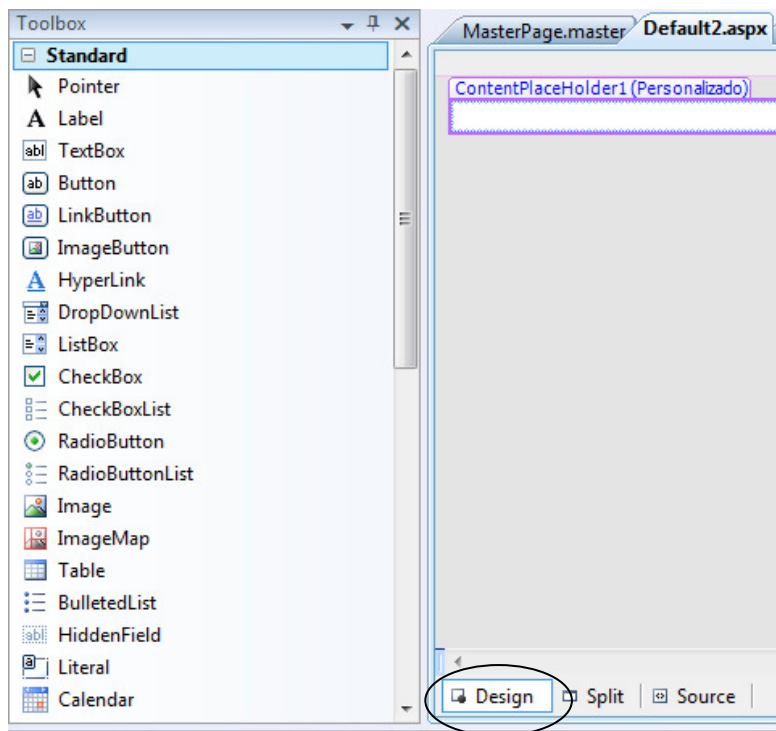
```
ImageUrl="~/Images/Waterlilies.jpg" runat="server"/>
```

Anidando Master Pages

Las Content Pages también pueden ser Master Pages. Ésto quiere decir que es posible derivar una Master Page a partir de otra Master Page. Por ejemplo, podríamos tener una Master Page de primer nivel que represente la cabecera/pie de página y la navegación global del sitio, y después Master Pages separadas que deriven de esta Master para definir los aspectos de las diferentes sub-secciones del sitio. Las Content Pages derivarán de la página maestra correspondiente a la sección a la que pertenece la Content Page.

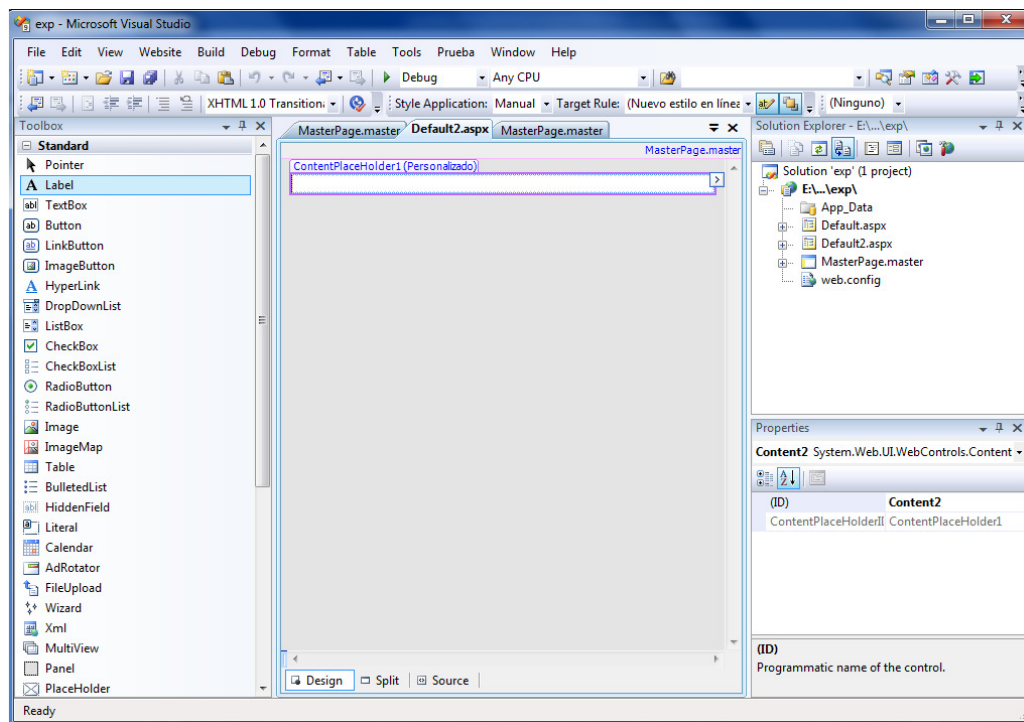


5.- CONTROLES EN UNA CONTENT.PAGE



La forma más fácil para insertar controles es desde design o diseño.

Vamos a distinguir las partes de la ventana:





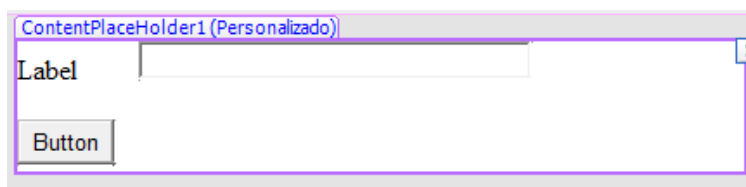
Desde la parte izquierda el cuadro de herramientas, en el centro el formulario principal (content.page) y en la parte derecha las ventanas de propiedades y el explorador de soluciones.

Los controles son los elementos que insertamos dentro de un formulario y que nos va a permitir interactuar entre el usuario y el código: botones, cuadros de texto, etiquetas, cuadros desplegables, cuadrículas de datos. En definitiva todos y cada uno de los elementos que vemos en los formularios de todas las aplicaciones. La lista de controles básicos disponibles la tenemos a la izquierda, en el panel de herramientas.

Empezando con la aplicación

Para añadir controles al formulario utilizaremos la barra de herramientas. Por ejemplo, para añadir una etiqueta (Label) y una caja de texto (TextBox), simplemente haremos doble-clic sobre esos elementos de la barra de herramientas y se añadirán al formulario o podemos también agarrarlos y arrastrarlos hasta nuestro formulario.

Para nuestro ejemplo, añadiremos un botón (Button) y lo situaremos debajo del cuadro de texto (Textbox). Luego añadimos una etiqueta (Label) para que quede de esta forma:

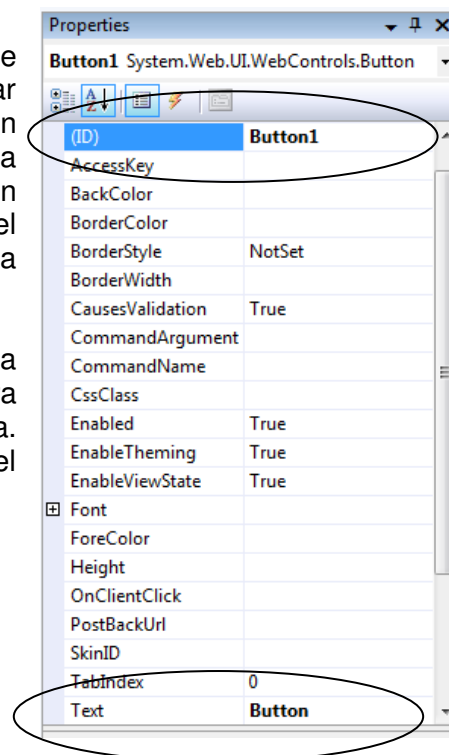


Como vemos, por defecto, el IDE pone un texto genérico (Label, Button, etc) y un nombre al control tal como label1, textbox1, button1, es decir utiliza el tipo de control y lo va numerando tantas veces como controles iguales tengamos en el formulario (label1, label2, label3,...). Es por ponerle un nombre inicial, ya que siempre los controles deben tener un nombre único.

Ahora vamos a cambiar el texto y el nombre que contiene el botón "Button". Para ambos hay que utilizar la ventana de propiedades (A la derecha-abajo), en esta ocasión el elemento que nos interesa de esa ventana de propiedades son Text e ID. Escribimos en ambos la palabra "Mostrar" y cuando pulses Intro o el tabulador veremos que el texto del botón se ha actualizado.

Hacemos lo mismo con la etiqueta "Label", recuerda que hay que seleccionarla primero haciendo clic para que se muestren las propiedades de la etiqueta. Escribimos "Nombre:" en Text y pulsamos intro o el tabulador.

Fíjate que en la parte superior de la ventana pone el control que estamos editando y de qué tipo es.





Escribir el código

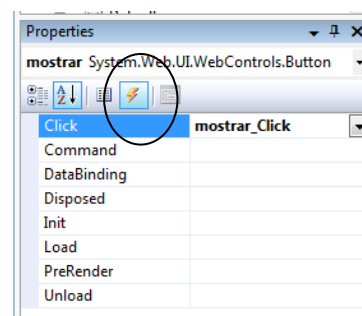
Ahora vamos a escribir el código que se ejecutará cada vez que se haga clic en el botón que hemos añadido. Para ello, selecciona el botón "Mostrar" y hacemos doble clic en él, se mostrará una nueva ventana, en este caso la ventana de código del botón, asociada con el formulario que tenemos en nuestro proyecto. Nos mostrará:

```
Default2 mostrar_Click(object sender, EventArgs e)
1 using System;
2 using System.Collections;
3 using System.Configuration;
4 using System.Data;
5 using System.Linq;
6 using System.Web;
7 using System.Web.Security;
8 using System.Web.UI;
9 using System.Web.UI.HtmlControls;
10 using System.Web.UI.WebControls;
11 using System.Web.UI.WebControls.WebParts;
12 using System.Xml.Linq;
13
14 public partial class Default2 : System.Web.UI.Page
15 {
16     protected void Page_Load(object sender, EventArgs e)
17     {
18     }
19
20     protected void mostrar_Click(object sender, EventArgs e)
21     {
22     }
23 }
24
25
```

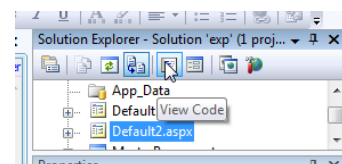
Aquí empieza la programación.

Que nosotros pulsemos doble clic en el botón y que aparezca un fragmento de código significa lo siguiente: VB .NET interpreta que quieres poner código para realizar alguna acción cuando se haga clic sobre el botón, por lo tanto te muestra ya el "procedimiento" o parte de código que se va a ejecutar cuando suceda esto. A esta forma de trabajar se le llama programación orientada a eventos. Es decir, cuando se produzcan el evento de pulsar el botón (click) ejecutas este código.

Por lo tanto vemos que los controles además de tener propiedades (que modifican su aspecto) también atienden a una serie de eventos (clic, doble clic). Los eventos a los que atienden los controles los podemos ver en la ventana de propiedades, seleccionando:



Hay otra forma de acceder a los eventos de los controles y es esta. Vete a la vista del código del formulario pulsando en el botón de código, que muestro en la imagen siguiente:






Escribimos el siguiente código en el hueco dejado por Visual Studio entre las líneas que hay entre { ... }

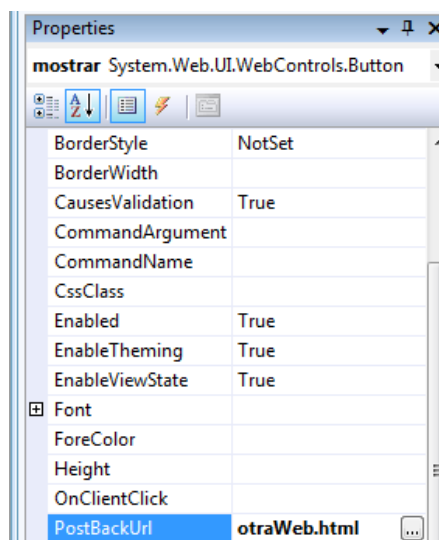
```
1 using System;
2 using System.Web;
3
4 public partial class Default2 : System.Web.UI.Page
5 {
6     protected void Page_Load(object sender, EventArgs e)
7     {
8
9     }
10    protected void mostrar_Click(object sender, EventArgs e)
11    {
12        HttpContext.Current.Response.Write (
13            "<h1>Pulsado mostrar "+ TextBox1.Text + "</h1>"
14        );
15    }
16 }
17
```

Pulsamos F5 para que se ejecute el código que hemos escrito o pulsa en el botón "play" que está en la barra de botones.

Cuando se presente el formulario escribe algo en el cuadro de texto, pulsa en el botón Mostrar y veremos que se muestra un texto inicial diciéndote "Pulsado mostrar" y a continuación lo que hayas escrito en el cuadro de texto (TextBox1):

Ya tenemos nuestra aplicación funcionando, creada con Visual Studio .NET.

Vamos ahora a añadir al botón la capacidad de saltar a otra Web. Cambiaremos la propiedad **PostBackUrl** poniendo la dirección a donde queremos que salte. Incluso con los  podremos elegir clicando de nuestro sitio web la página a la que saltará.





6.- EJERCICIO

Se entrega un zip con los ficheros que se ven en la imagen.

Se pide transformar `_master.html` en una master page siguiendo la estructura de carpetas que se ve en la imagen.

Se pide crear 2 páginas “content pages” ambas con código asociado. Llámalas `default.aspx` y `rdefault.aspx`.

(Nota: estos ejercicios servirán de base para otros, en caso de no acabarlos os lo proporcionaríamos).

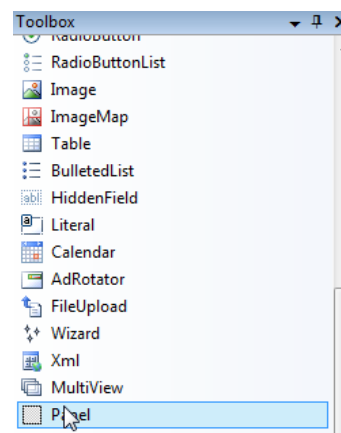
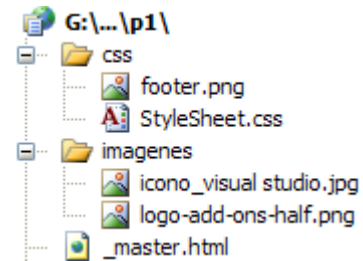
Default.aspx

La página está entera con controles de servidor.

Todos los controles están agrupados con un panel. Cada control lleva su etiqueta asociada.

Campos:

- De texto para el “TxtNombre”.
- Lista desplegable “LisRegalo”. Las opciones son: “” (en blanco), “Inmobiliario”, “Servicio”, “Transporte”.
- De texto “txtFecha1”.
- De texto “txtFecha2”.
- De texto “txtDni”.
- Un grupo de botones de radio “radTipo” con las opciones de “Hombre funcionario”, “Mujer funcionaria” y “Otros”
- Por último un campo `txtOtros`, para cuando en la pregunta anterior se contesta “Otros”.
- Un botón de servidor con `PostBackUrl="~/paginas/rDefault.aspx"`



RECOGIDA DE DATOS

Nombre

Regalo

Fecha entrega

Fecha entrega alternativa

D.N.I.

Indicar sexo y condición

Hombre Funcionario

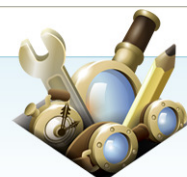
Mujer funcionaria

otros

En caso de marcar otros señale que otros

Curso de programación c# en .NET

En este ejemplo vamos a crear una página web que recoja información sobre personas pertenecientes a la comunidad universitaria. Después se hará sorteo para regalar unos bolis.





rDefault.aspx

Esta página recogerá los valores que se envían.

```
<%@ Page Language="C#"
MasterPageFile="~/MasterPage.master"
AutoEventWireup="true"
CodeFile="rDefault.aspx.cs"
Inherits="_rDefault"
Title="Página de respuesta" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="CPHolder1" Runat="Server">
<asp:Panel ID="Panel1" runat="server" GroupingText="OBTENCIÓN DE DATOS" CssClass="tit">
    <%
        foreach (string control in Request.Form)
        {
            Response.Write("<p>" + control + " " + Request.Form[control] + "</p>");
        }
    %>
</asp:Panel>
</asp:Content>
```

OBTENCIÓN DE DATOS

```
__EVENTTARGET
__EVENTARGUMENT
__VIEWSTATE /wEPDwUJNJE4MjU5NDk3ZGSHqKQPkhaT6X3ewhvm6M6fg9ncsRQ==
__PREVIOUSPAGE 2tuk9WlroXXqk64LGDYhp2wW8H78hyq3dsRaotWSFAA1
__EVENTVALIDATION
/wEWDwK+mbOeBQLqutGmCwLJmZ6VBgkE7LTLBAKy7ZhWAovakOkLAtqShdMIar+p5700AqGvmZgEauXij4UDAov48qUBAon1zMcoAr
/Y/+wOAr+Sk7cOAOdiyWMyMphQruEMa/Hi6QBGq2IT3/WnA==
ctl00$ContentPlaceHolder1$txtNombre Marcos
ctl00$ContentPlaceHolder1$LisRegalo Inmobiliario
ctl00$ContentPlaceHolder1$btFecha1 10/10/2010
ctl00$ContentPlaceHolder1$btFecha2
ctl00$ContentPlaceHolder1$btDni 214444444
ctl00$ContentPlaceHolder1$RadioButtonList1 Hombre Funcionario
ctl00$ContentPlaceHolder1$btOtros xx
ctl00$ContentPlaceHolder1$Button1 Button
```

Curso de programación c# en .NET

En este ejemplo vamos a crear una página web que recoja información sobre personas pertenecientes a la comunidad universitaria. Después se hará sorteo para regalar unos bols.

