Hada T4: Tecnologías web

Tecnologías web.

Objetivos

- Aprender las tecnologías fundamentales de programación web
- HTML
- CSS
- ASP.NET
- Programación de aplicaciones web con visual studio .NET

HTML

- **HTML**
- ASP . NET
- CSS

HTML

- Hypertext Markup Language
- **Hypertext**
    - Es texto ordinario al que se le incorporan funcionalidades adicionales como el formato, imágenes, multimedia y enlaces a otros documentos.
- **MarkUp**
    - **Es el proceso de tomar el texto ordinario e incorporar símbolos adicionales.** Cada uno de estos símbolos identifica a un comando que le indica al navegador cómo mostrar ese texto.

Elementos HTML

- Los elementos son los componentes fundamentales de HTML
- Cuentan con dos propiedades básicas:
    - **Atributo**
    - **Contenido**
- En general se conforman con una *Etiqueta* de apertura y otra cierre
- Los **atributos** se colocan dentro la etiqueta de apertura, y el **contenido** se coloca entre la etiqueta de apertura y la de cierre

Atributos de elementos

- Los atributos de un elemento son pares de nombres y valores separados por un ‘=‘ que se encuentran dentro de la etiqueta de apertura de algún elemento. Los valores deben estar entre comillas

```
<span id=‘iddeesteelemento' style='color:red;' title=‘Curso de HTML'>
  Curso de HTML
</span>
```

Estructura de página HTML

```
<HTML>
    <HEAD>
    <TITLE> Mi web! </TITLE>
    </HEAD>
    <BODY>
        <H1> Hello World </H1>
        <! Rest of page goes here. This is a comment. >
    </BODY>
</HTML>
```

Elementos principales

- Etiquetas que definen la estructura del documento
- **<HTML>... </HTML>**

  Delimita el Documento HTML
- **<HEAD> ... </HEAD>**

  Delimita el encabezado del Documento HTML

  En general incluye los metadatos del documento y Scripts.
- **<BODY> ... </BODY>**

  Delimita el Cuerpo del Documento HTML.

  Es donde se incluyen los contenidos visibles del documento.

Algunos elementos posibles en HEAD

- **<TITLE> … </TITLE>**

  Define el título del documento HTML

- **<SCRIPT> … </SCRIPT>**

  Se utiliza para incluir programas al documento. En general se tratan de Javascripts.

- **<STYLE> … </STYLE>**

  Especifica un estilo CSS para ser utilizado en el documento.

- **<META> … </META>**

  Permite especificar información de interés como: autor, fecha de publicación, descripción, palabras claves, etc.

Más etiquetas

Bloques de texto

<h1> ..<h6>

<p>

<blockquote>

<address>

<pre>

Características texto

<b>

<sub>

<sup>

<small>

Listas

<dl>

<dt>

<dd>

Scripts

<script>

Semánticos

<section>

<article>

<header>

<footer>

<nav>

<aside>

<figure>

<figcaption>

<summary>

Imágenes y mapas

<map>

<area>

Tablas

<table>

<tr>

<td>

<th>

Formularios

<form>

<input>

<select>

<option>

<textarea>

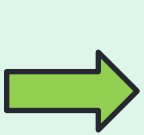
Enlaces

<a>

Listas no ordenadas (1)

- Listas no ordenadas …
- Para añadir un elementos a una lista utilizaremos la etiqueta …

```
<ul>
    <li> Azul </li>
    <li> Amarillo </li>
    <li> Rojo </li>
    <li> Verde </li>
</ul>
```



- Azul
- Amarillo
- Rojo
- Verde

Listas no ordenadas (2)

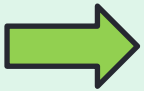
- El atributo ***type***, que permitía cambiar el tipo de marca *(type="circle", type="square", type="disc"*), no forma parte de HTML5 (uso de CSS), aunque sigue siendo compatible

```
<ul>
    <li type="circle"> Azul </li>
    <li type="square"> Amarillo </li>
    <li type="disc"> Rojo </li>
</ul>
```

- Azul
- Amarillo
- Rojo

- Utilización de CSS (Style)

```
<ul>
    <li style="list-style-type:circle"> Azul </li>
    <li style="list-style-type:square"> Amarillo </li>
    <li style="list-style-type:disc"> Rojo </li>
</ul>
```



- Azul
- Amarillo
- Rojo

Listas ordenadas (1)

- Listas ordenadas …
- Para añadir un elementos a una lista utilizaremos la etiqueta …

```
<ol>
    <li> Azul </li>
    <li> Amarillo </li>
    <li> Rojo </li>
    <li> Verde </li>
</ol>
```

1. Azul
2. Amarillo
3. Rojo
4. Verde

Listas ordenadas (2)

- El atributo **type**, permite modificar el tipo de numeración (type="1") por letras (type="a") o números romanos (*type="i"*).

```
<ol type="a">
    <li> Azul </li>
    <li> Amarillo </li>
    <li> Rojo </li>
    <li> Verde </li>
</ol>
```

a. Azul
b. Amarillo
c. Rojo
d. Verde

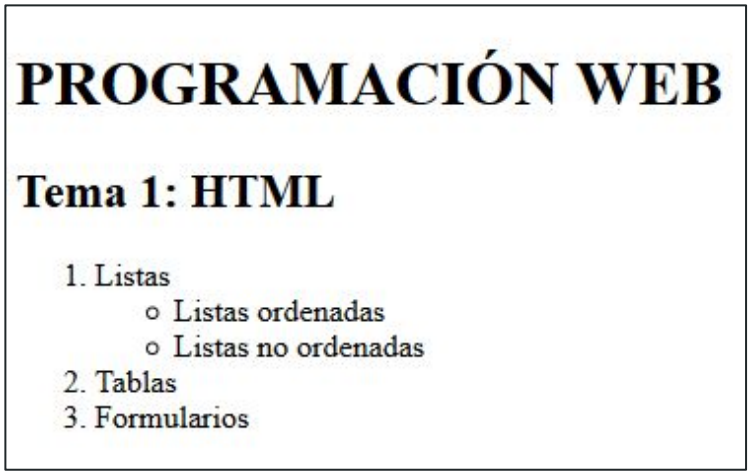
- También es posible modificar la lista para que el primer punto empiece por otro número (tipo de numeración de números), utilizando el atributo **start** en ol (start=*"25", etc.).*

```
<ol start="25">
    <li> Azul </li>
    <li> Amarillo </li>
    <li> Rojo </li>
    <li> Verde </li>
</ol>
```

25. Azul
26. Amarillo
27. Rojo
28. Verde

Ejercicio 1

- Crea una página web que tenga el mismo aspecto que la siguiente imagen



Codigo...

```
<HTML>
    <HEAD>
    <TITLE> Ejercicio 1 HADA </TITLE>
    </HEAD>
    <BODY>
      <!lista ordenada>
        <!lista desordenada>
    </BODY>
</HTML>
```

Enlaces

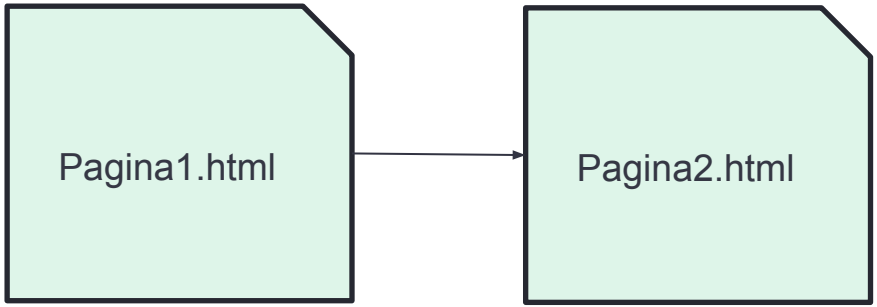
- La etiqueta <a>…, inserta un enlace.

```
<a href=“destino_del_enlace”>Texto del enlace </a>
```

- El destino del enlace puede ser a
  - Un lugar de la página en curso
  - Otra página del sitio web
  - Algún lugar de otra página del sitio web
  - Una página de otro sitio existente en la web
  - Una dirección de correo electrónico
  - Una archivo para descargas

Enlaces (2)

Enlace a otra página -- Cómo sería???



El código del enlace en la página 1 es:

Tablas (1)

- Creación de una tabla <table>…</table>
- Filas <tr>…</tr>
- Celdas <td>…</td>
- Celdas cabecera <th>…</th>
- Antes de aparecer CSS, las tablas eran fundamentales. **Hoy en día no se recomienda su uso para organizar el diseño.**
- Usar CSS.
- Combinación de celdas: colspan, rowspan

```
<table>
  <tr>
    <th> Asignatura </th>
    <th> Créditos </th>
  </tr>
  <tr>
    <td> Programación 1</td>
    <td> 6 </td>
  </tr>
  <tr>
    <td> Base de datos </td>
    <td> 6 </td>
 </tr>
</table>
```

Asignatura	Créditos
Programación 1	6
Base de datos	6

Tablas (2)

Ejemplo de tabla

```
        <body>
                <table>
                        <tr>
                                <th>Código</th>
                                <th>Municipio</th>
                        </tr>
                        <tr>
                                <td>1</td>
                                <td>Agres </td>
                        </tr>
                        <tr>
                                <td>2</td>
                                <td>Alcoy </td>
                        </tr>
                        <tr>
                                <td>3</td>
                                <td>Torremanzanas </td>
                        </tr>
                </table>
        <body>
</html>
```

Código	Municipio
1	Agres
2	Alcoy
3	Torremanzanas

Tablas (2)

Ejemplo de tabla utilizado css

```
<!DOCTYPE html>
        <head>
                <title> Ejemplo de tabla con CSS</title>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

                <style>

                        table { text-align: center;
                                border: 1px solid black;
                                border-collapse: collapse;
                                width: 300 px; }
                        td {border: 1px solid black;
                            background-color: #99ccff; }
                </style>
        </head>
```

Código	Municipio
1	Agres
2	Alcoy
3	Torremanzanas

Para la tabla (table):

- **Text-align**: alineación de las celdas de la tabla
- **Border**: borde de la tabla
- **Border-collapse**: para eliminar el espacio entre el borde de la tabla y los bordes de las celdas
- **Width**: ancho de la tabla

Para las celdas (td):

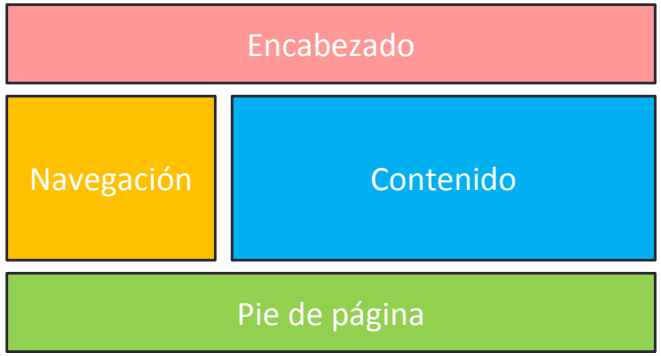
- **Border**: borde de la celda
- **Background-color**: color del fondo de la celda

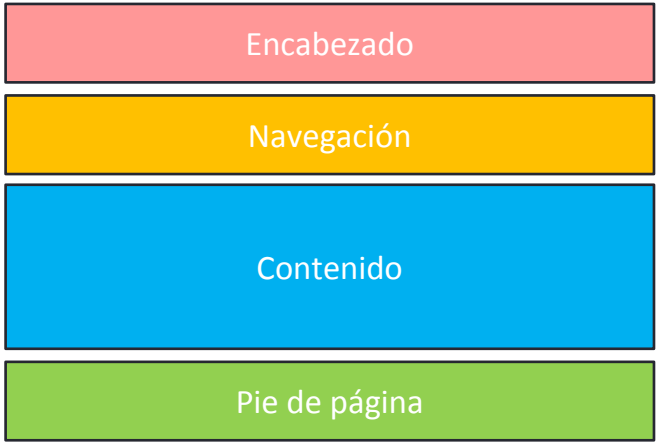
Etiquetas de organización

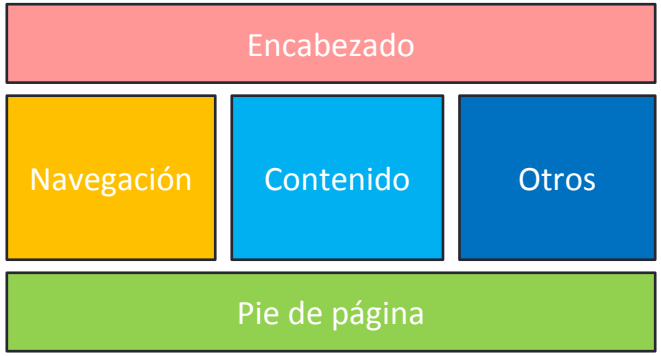
- Son etiquetas que permiten formar bloques, que describen mejor las partes de un documento
- Todas las web están formadas por:
    - Encabezado de página
    - Herramientas de navegación
    - Contenido
    - Zona anexa de elementos asociados al contenido (publicidad)
    - Pie de página

Etiquetas de organización (2)

Ejemplos de organización en una web







Etiquetas de organización (3)

- **<header> … </header>**

  Agrupa los elementos del encabezado de la página

- **<nav> … </nav>**

  Indica los elementos del menu de navegación

- **<footer> … </footer>**

  Agrupa los elementos del pie de página

- **<aside> … </aside>**

  Indica que se tratan de elementos anexos al contenido

Etiquetas de organización (3)

- **<section> … </section>**

Indica que una parte del contenido de la página se refiere a un tema en concreto

- **<article> … </ article>**

Define un contenido del documento que posee una identidad independiente dentro de la página (artículo de un blog, un post en un foro, o un product en la web de un comercio)

- **<figure> … </figure>**
  Permite incorporar una sección de imagen.
- **<figcaption> … </figcaption>**
  Contiene el texto explicativo de una imagen asociada

Etiquetas de organización (4)

Ejemplo

```
<!DOCTYPE html>
<html>
	<head>
		<title> Nuevas secciones en HTML 5 </title>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
		<link rel="stylesheet" href="css/estilo.css">
	</head>

	<body>
		<header>
			<img src="img/logo-ua.jpg" width="189" height="57" alt="logo"  />
			<h1>EJEMPLO SENCILLO EN HTML5</h1>
		</header>

		<nav>
			<ul>
				<li> <a href="">Inicio</a> </li>
				<li> <a href="">Elemento 1</a> </li>
				<li> <a href="">Elemento 2</a> </li>
				<li> <a href="">Elemento 3</a> </li>
				<li> <a href="">Contacto </a> </li>
			</ul>
		</nav>

		<section>
			<h1> Asunto de la página </h1>

			<figure>
				<img src="img/planoUA.jpg" width="720" height="495" alt="foto"  />
				<figcaption>Figura 1: plano de la Universidad de Alicante</figcaption>
			</figure>

			<article>
				<p>Lorem ipsum dolor sit amet consectetur adipiscing elit ornare suscipit, lectus iacu
placerat potenti sagittis cubilia. Nullam fames a nec quis molestie, ad platea ornare phasellus metus bibendum, himena
senectus interdum aliquam facilisi fermentum volutpat et justo, sagittis curae phasellus platea purus proin eros. </p>
			</article>
		</section>

		<aside>
			<h3>Archivos</h3>
			<ol>
				<li> <a href="">Datos_2015</a> </li>
				<li> <a href="">Datos_2016</a> </li>
				<li> <a href="">Datos_2017</a> </li>
				<li> <a href="">Datos_2018</a> </li>
				<li> <a href="">Contacto </a> </li>
			</ol>
		</aside>

		<footer>
			Universidad de Alicante 2018 <a href="http://www.ua.es">www.ua.es</a>
		</footer>

	</body>
</html>
```

Utilizando un CSS, le daríamos el estilo adecuado a la web

```
html,body {
	height:100%;
}

h1 {
	text-align: center;
}

header {

	display:block;
	background:#086af0;
	padding:6px 10px;
	color:white;

}
section {
	width: 79%;
	background: #ccc;
	float: left;
	overflow: auto;
	padding-bottom: 60px;
	padding-top:30px;
}

aside {
	float: right;
	border: 1px solid red;
	width: 19%;
	border: 1px solid red;
}
footer {
	position: relative;
	margin-top: -50px;
	height: 40px;
	padding:5px 0px;
	clear: both;
	background: #286af0;
	text-align: center;
	color: white;
}
figure {
	display: table; margin: 0 auto;
}
```

```
nav {
	position : absolute;
	left : 0;
	width : 100%;
	height: 50px;
	background-color: #333;
overflow: hidden;
	color: white;
}
nav ul {
	margin : 0 auto;
	width : 940px;
	list-style : none;
}
nav ul li {
	float : left;
}

nav ul li a {
	display : block;
	margin-right : 20px;
	width : 140px;
	font-size : 18px;
	line-height : 44px;
	text-align : center;
	text-decoration : none;
	color : white;
}
nav ul li a:hover {
	color : #fff;
}
nav ul li.selected a {
	color : #fff;
}
```

Formularios

- Recaba la información introducida por el usuario
- La información es enviada a una página (normalmente al servidor) después de pulsar en un botón
- El servidor recibe la información como pares de datos, con cada valor asociado a cada nombre de control
- El servidor procesa dichos datos, y responde el resultado
- Nosotros utilizaremos los formularios de ASP. NET
- Se define mediante la etiqueta <form> … </form>

Formularios (2)

Atributos de la etiqueta form

- name: asignar un nombre al formulario
- action: acción que debe realizar (web, correo, programa)
- enctype: especifica el formato (application/x-www-form-urlencoded)
- method: el método para mandar la información

```
<!DOCTYPE html>
        <head>
                <title> Ejercicio formulario 1</title>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        </head>

        <body>
                <form method="get" action="ejercicio2.html">
                        <label for="nombre">Nombre:</label>
                        <input type="text" id="nombre" size="20"> <br>
                        <label for="username">Apellidos:</label>
                        <input type="text" id="apellidos" size="40"> <br>
                        <button>Enviar</button>
                </form>

        <body>
</html>
```

Formularios (3)

- Dos opciones para enviar la información (method):
  - GET. Los datos se envían (caracteres ASCII) dentro de la propia URL, unidos por &

http://www.miweb.com/index.aspx?id=1&nombre=pepe&apellidos=perez

Para enviar poco texto, datos que no requieran de seguridad, sencillez

  - POST. Los datos no se envía en la URL y son invisibles para el usuario

http://www.miweb.com/index.aspx

Para enviar grandes campos de texto, imágenes, URL más legibles, seguridad de información

Formularios (4)

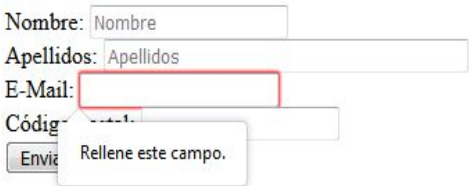
Campo de texto

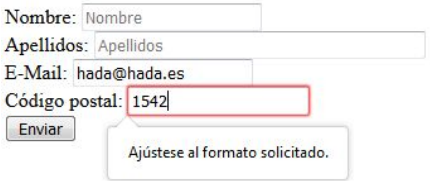
<input type="text">

Algunos atributos

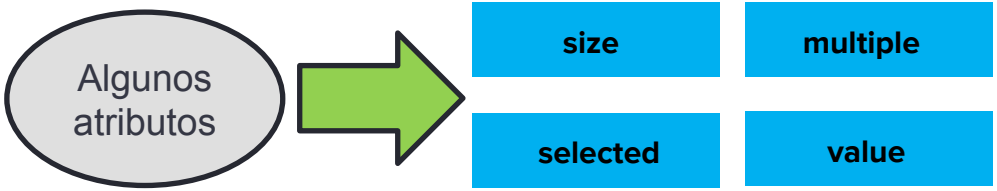
- maxlength
- size
- readonly
- required
- placeholder
- autofocus
- Pattern

```
<form method="get" action="ejercicio2.html">
        Nombre:
        <input type="text" name="nombre" placeholder="Nombre" size="20"> <br>
        Apellidos:
        <input type="text" name="apellidos" placeholder="Apellidos" size="40"> <br>
        E-Mail:
        <input type="text" name="mail" required> <br>
        Código postal:
        <input type="text" name="postal" pattern="[0-9]{5}"> <br>

        <button>Enviar</button>
</form>
```





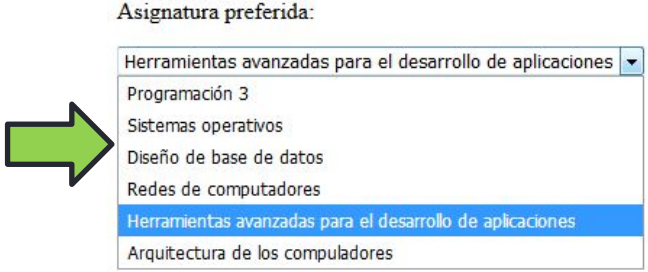
Formularios (5)



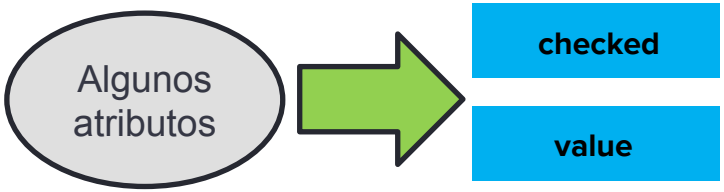
Lista desplegable

- <select> … </select> indica el uso de una lista desplegable
- <option> …</option> forma los elementos de la lista

```
<!DOCTYPE html>
   <head>
      <title> Ejercicio formulario 3</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   </head>

   <body>
      <form action="">
         <p>Asignatura preferida: </p>
         <select>
            <option value="PR1">Programación 3 </option>
            <option value="SIO">Sistemas operativos</option>
            <option value="FDB">Diseño de base de datos</option>
            <option value="RED">Redes de computadores</option>
            <option value="HAD" selected>Herramientas avanzadas para el desarrollo de aplicaciones</option>
            <option value="ARC">Arquitectura de los compuladores</option>
         </select>
      </form>
   <body>
</html>
```



Formularios (6)



Botones de selección única

<input type="radio">

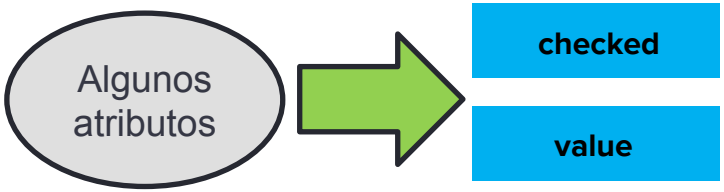
```
<!DOCTYPE html>
   <head>
      <title> Ejercicio formulario 4</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   </head>

   <body>
      <form action="ejercicioFormulario4.html">
         <p>Sexo:</p>
         <input type="radio" name="sexo" value="H">Hombre <br>
         <input type="radio" name="sexo" value="M">Mujer  <br>
         <button>Enviar</button>
      </form>
   <body>
</html>
```

- Cuando pulsemos en el botón *Enviar*, se enviarán los datos con el valor establecido con *value*.

ejercicioFormulario4.html?sexo=H

ejercicioFormulario4.html?sexo=M

Formularios (7)



Botones de selección múltiple

<input type="ckeckbox">

```
<!DOCTYPE html>
   <head>
      <title> Ejercicio formulario 5</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   </head>

   <body>
      <form action="ejercicioFormulario5.html">
         <p>Selección de asignaturas matriculadas:</p>
         <input type="checkbox" name="a1">Sistemas operativos <br>
         <input type="checkbox" name="a2" checked>Diseño de base de datos <br>
         <input type="checkbox" name="a3">Redes de computadores <br>
         <input type="checkbox" name="a4" checked>Herramientas avanzadas para el desarrollo de aplicaciones <br>
         <input type="checkbox" name="a5">Arquitectura de los compuladores <br>
         <button>Enviar</button>
      </form>
   <body>
</html>
```

Selección de asignaturas matriculadas:

- [ ] Sistemas operativos
- [x] Diseño de base de datos
- [ ] Redes de computadores
- [x] Herramientas avanzadas para el desarrollo de aplicaciones
- [ ] Arquitectura de los compuladores

Enviar

- Cuando pulsemos el botón *Enviar*, se enviarán los datos con el valor a on de cada opción seleccionada.
- Si tiene el atribute value reemplazará el on por el valor asociado.

ejercicioFormulario5.html?a2=on&a4=on

ejercicioFormulario5.html?a2=on&a4=HADA

Formularios (8)

Otras elementos del formulario

- Botón de envio
- Botón de anulación
- Área de texto
- Campos ocultos
- Contraseñas
- Campos de transferencia de archivos
- Etiquetas de campos
- Campo de texto URL
- Campo de texto con lista de sugerencia
- Campo de texto para dirección de correo
- Campo de texto numérico
- Campo de texto de búsqueda
- Campo de texto de fecha
- Campo de texto de color
- Cursores

ASP .NET

- HTML
- **ASP . NET**
- CSS

Índice

Introducción a las aplicaciones web ASP.NET

1

Aplicaciones Web ASP.NET

- Combinación de archivos, páginas, manejadores, módulos y código ejecutable que puede invocarse desde un **directorio virtual.**
- Se dividen en varias páginas web.
- Comparten un conjunto de recursos y opciones de configuración común.
- Cada aplicación tiene su propio:
  - Conjunto de caché
  - Datos de estado de sesión

Dónde se guarda la aplicación??

➔ Directorio Virtual

- Una aplicación web sólo existe en una localización que ha sido publicada por IIS como un **Directorio Virtual**.

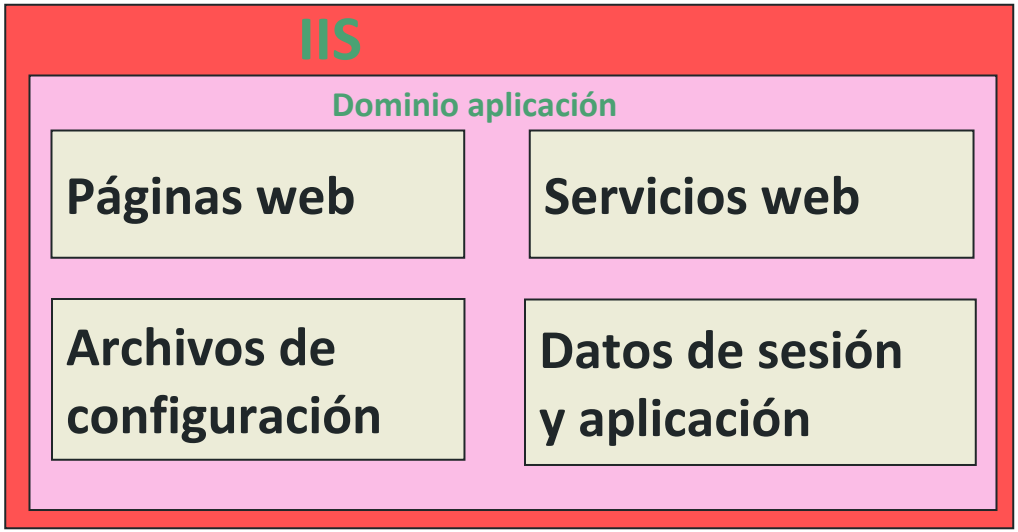
- Un **directorio virtual** es un recurso compartido identificado por un alias que representa la localización física en el servidor.

- **//localhost** es la carpeta virtual raíz del ordenador (**\InetPub\wwwroot**)

Directorio virtual

- Directorio virtual: estructura de agrupación básica que delimita una aplicación.

- Creación y administración desde IIS (Internet Information Server)



En VisualStudio…
Sitio Web o Proyecto Web?

- Sitio Web: conjunto de páginas Web independientes.
  - Para páginas Web sencillas (ej, página Web personal…)

- **Proyecto Web:** conjunto de páginas Web enlazadas con un archivo de proyecto.
  - Para aplicaciones avanzadas
  - Podemos referenciar DLLs, etc

Necesitamos Internet Information Server?

- Visual Studio dispone de su propio servidor de desarrollo, por lo que para hacer pruebas en nuestro ordenador no necesitamos tener instalado IIS.
- Sin embargo, para poder desplegar nuestra aplicación en un servidor, si lo necesitaríamos.

En Visual Studio…
Dónde se guarda la aplicación?

- File system
- C:\Documents and Settings\aaaa\Mis documentos\Visual Studio 2005\WebSites\WebSite2
  - http://localhost:3371/WebSite2/Default.aspx

- Local IIS: Carpeta del sitio web por defecto (http://localhost)
  - Por ejemplo C:\Inetpub\wwwroot\WebSite1

Code-inline vs Code-behind

- “Etiquetas” declarativas
  - HTML, controles del servidor, texto estático
- A diferencia de ASP, buena separación entre el código y las etiquetas

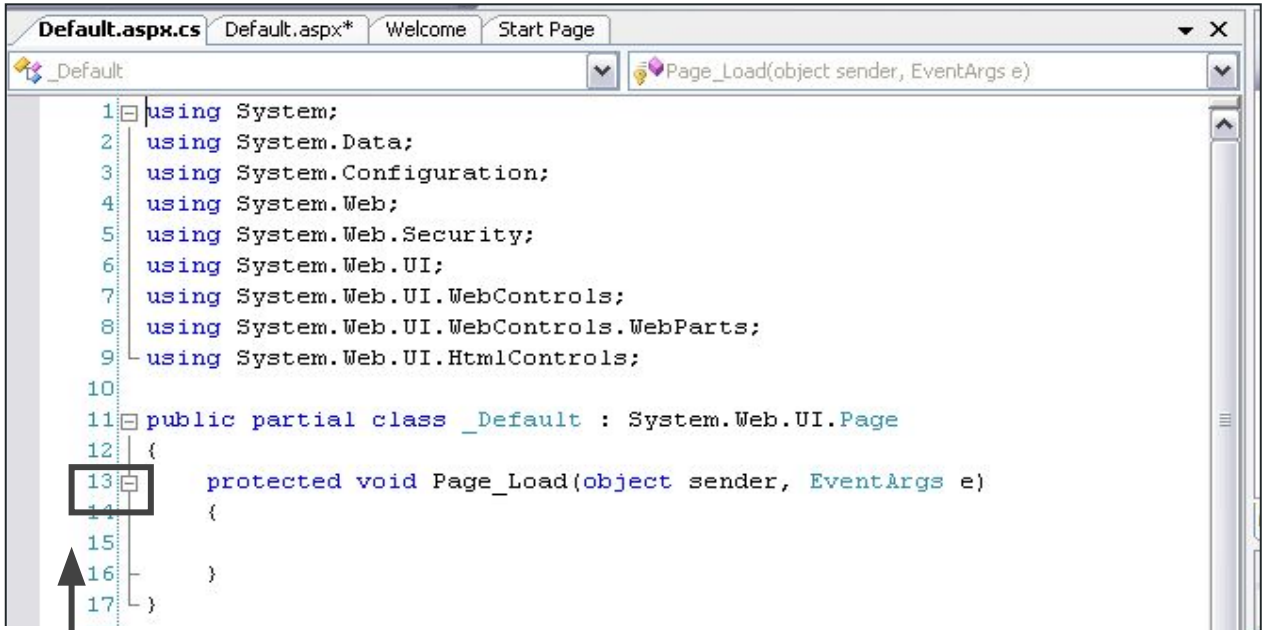
Code-behind

- El código para manejar eventos se sitúa en un **archivo físico separado** de la página que contiene los controles de servidor y las etiquetas.
  - Extensión *.aspx*
  - Extensión *.cs*, *.vb* … (*code-behind*)

- UTIL mantenerlo por separado :

  Es común en grupos de proyectos tener
  - **diseñadores** trabajando en la IU de la aplicación
  - y **desarrolladores** en el comportamiento o código.

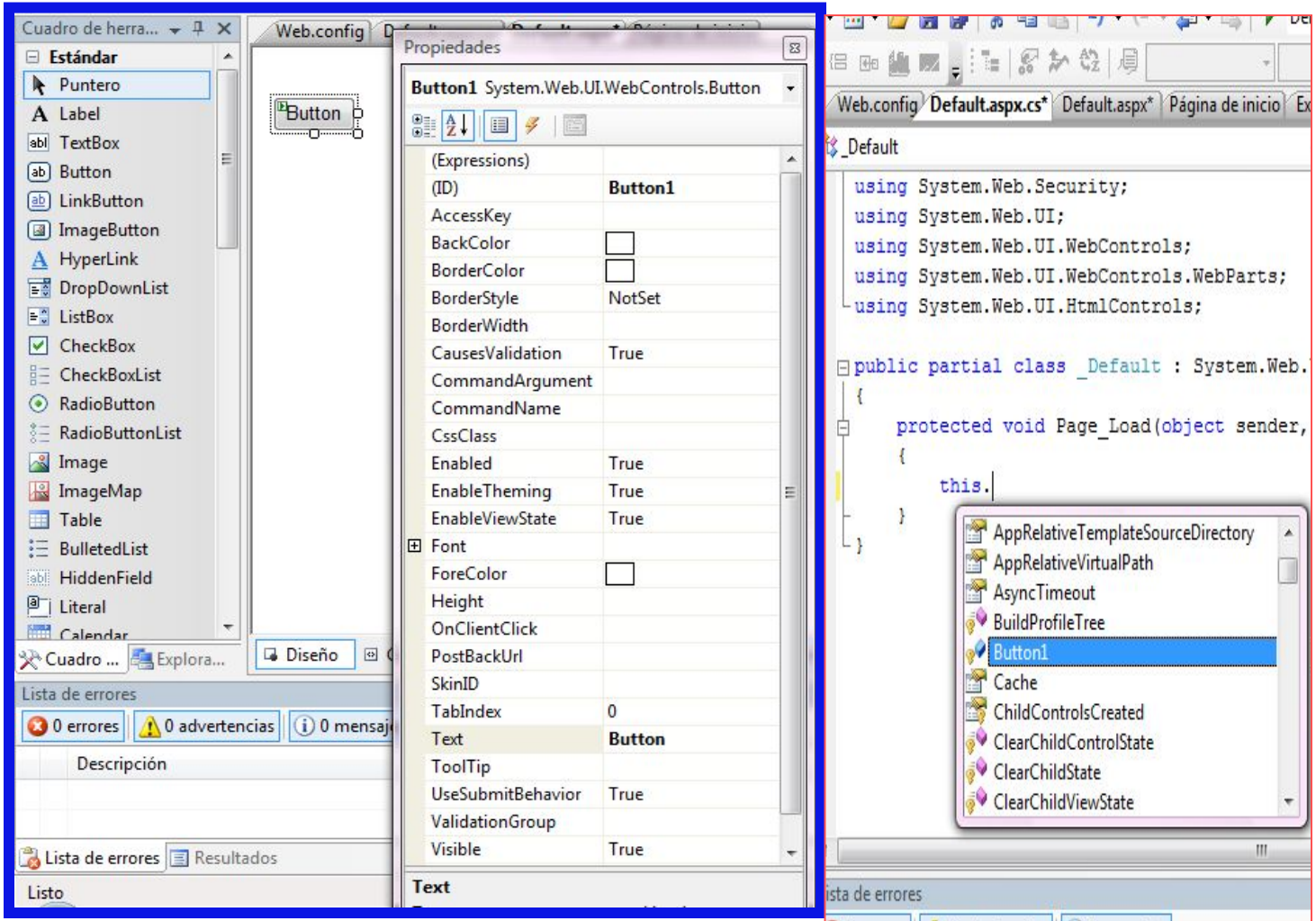
Ver código (C#)



Expander, contraer…

Definición de una clase parcial

Procedimiento que responde al evento LOAD de la página


Cuadro de herra...
Estándar
Puntero
Label
TextBox
Button
LinkButton
ImageButton
HyperLink
DropDownList
ListBox
CheckBox
CheckBoxList
RadioButton
RadioButtonList
Image
ImageMap
Table
BulletedList
HiddenField
Literal
Calendar
Cuadro ...
Explora...
Web.config
Button
Diseño
Propiedades
Button1 System.Web.UI.WebControls.Button
(Expressions)
(ID) Button1
AccessKey
BackColor
BorderColor
BorderStyle NotSet
BorderWidth
CausesValidation True
CommandArgument
CommandName
CssClass
Enabled True
EnableTheming True
EnableViewState True
Font
ForeColor
Height
OnClientClick
PostBackUrl
SkinID
TabIndex 0
Text Button
ToolTip
UseSubmitBehavior True
ValidationGroup
Visible True
Text
Lista de errores
0 errores
0 advertencias
0 mensaj
Descripción
Lista de errores
Resultados
Listo
Web.config
Default.aspx.cs*
Default.aspx*
Página de inicio
_Default
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class _Default : System.Web.
{
protected void Page_Load(object sender,
{
this.
}
}
AppRelativeTemplateSourceDirectory
AppRelativeVirtualPath
AsyncTimeout
BuildProfileTree
Button1
Cache
ChildControlsCreated
ClearChildControlState
ClearChildState
ClearChildViewState
ista de errores

Formulario ASP.NET

2

Formularios Web (I)

- Técnicas para Rápido Desarrollo de Aplicaciones (RAD)
- Podemos:
  - Arrastrar y soltar controles en un formulario
  - Escribir el código soporte
- La aplicación se desarrolla para un servidor web
- Los usuarios interactúan con la aplicación a través de un navegador
- Proporcionan una aproximación orientada a:

Objetos	Eventos	Gestión de estado

  - pueden ejecutarse, virtualmente, sobre cualquier navegador compatible con HTML.
- **Programación del lado del servidor para manejar eventos del lado del cliente**

Formularios Web (III)

- Todos los controles de servidor deben aparecer dentro de una etiqueta <form>, y esta etiqueta tiene que contener el atributo **runat="server".**
- Este atributo indica que el formulario se debe procesar en el servidor.
- También indica que los controles que contiene pueden ser accedidos por scripts del servidor:

```
<form runat="server">
    ...HTML + controles de servidor
</form>
```

- **Una pági** ...unat="server">

Aplicación web utilizando Web Forms

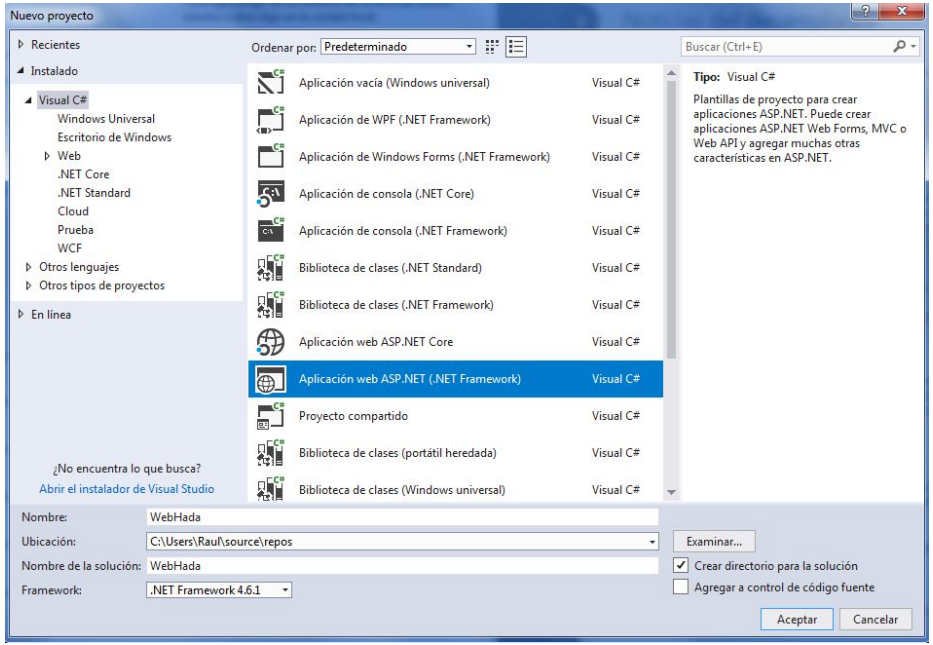
- **Vamos a crear nuestra primera aplicación web ASP.NET utilizando Web Forms.**
- Para ello utilizaremos **Visual Studio**, de manera que crearemos una aplicación web ASP.NET, y un formulario que permita al usuario introducir sus datos personales, validar los datos y si todo está correcto, cuando pulse en el botón *Enviar datos*, se enviarán los datos a un servidor.
- En el formulario web el usuario podrá introducir la siguiente información: **Nombre, apellidos, dirección, código postal, sexo y correo electrónico**

Aplicación web utilizando Web Forms

Creación del proyecto

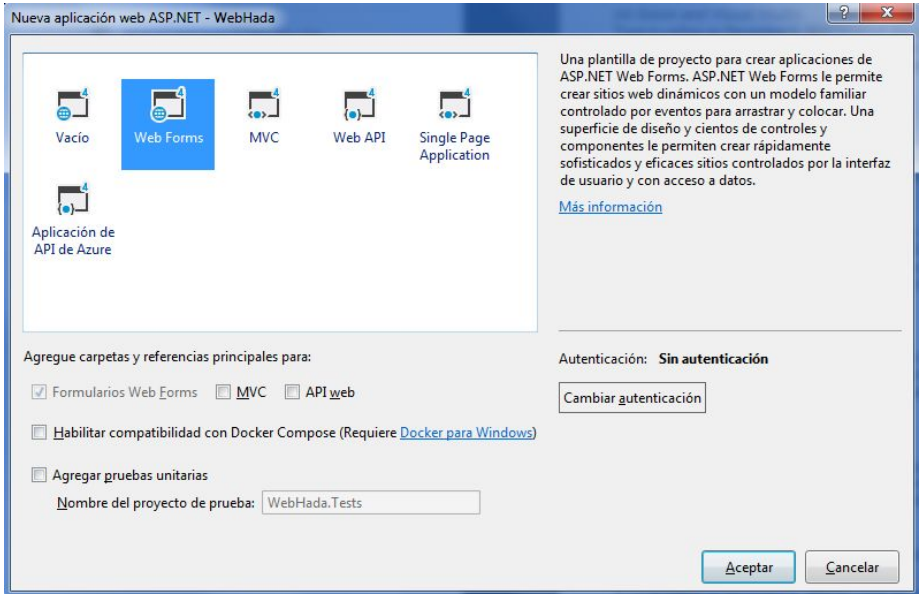
- Para crear el proyecto desde el entorno de Visual Studio, hay que acceder al menú *Archivo -> Nuevo -> Proyecto.*

- Dentro del menú *Visual C#*, acceder al menú *Web* y seleccionar como tipo de aplicación **Aplicación web ASP.NET(.NET Framework)**.
- Damos un nombre al proyecto, en este caso ***WebHada***.



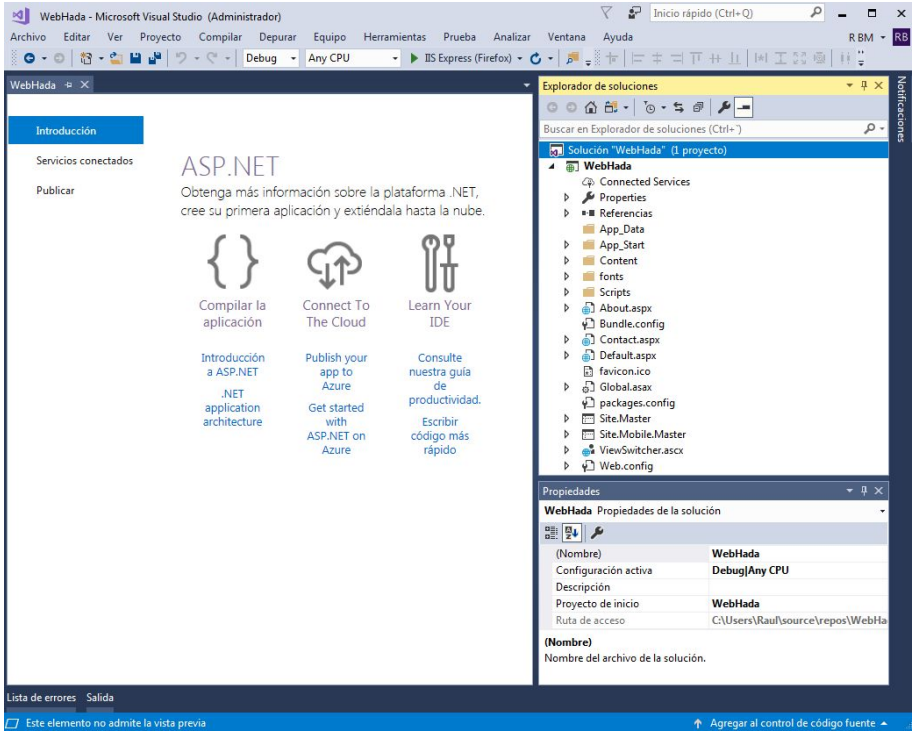
Aplicación web utilizando Web Forms

- A continuación, aparece la pantalla para seleccionar la plantilla en que se basa la aplicación web. En este caso seleccionamos *Web Forms*.
- Al pulsar en *Aceptar*, se creará el proyecto.
- Hay que tener en cuenta, que este proyecto se creará dentro de una solución con el mismo nombre que el proyecto.



Aplicación web utilizando Web Forms

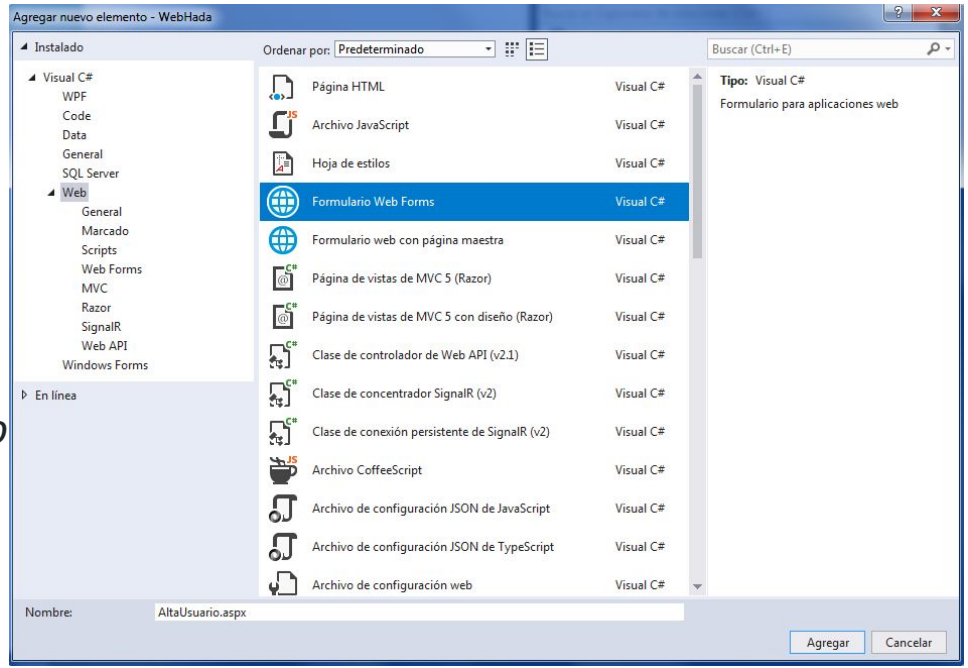
- Ya tenemos creado el proyecto base a partir del cual trabajaremos
- El explorador de soluciones, muestra el contenido de la solución (proyectos de la solución y elementos de cada proyecto).
- Propiedades de los elementos seleccionados



Aplicación web utilizando Web Forms

Creación del formulario

- Desde el explorador *de soluciones,* pulsamos con el botón derecho del ratón sobre el nombre del proyecto WebHADA.
- Aparece un menú contextual, del que elegiremos la opción *Agregar -> Nuevo elemento*.
- Elegiremos la opción **Formulario Web Forms,** y le daremos el nombre de *AltaUsuario.aspx*



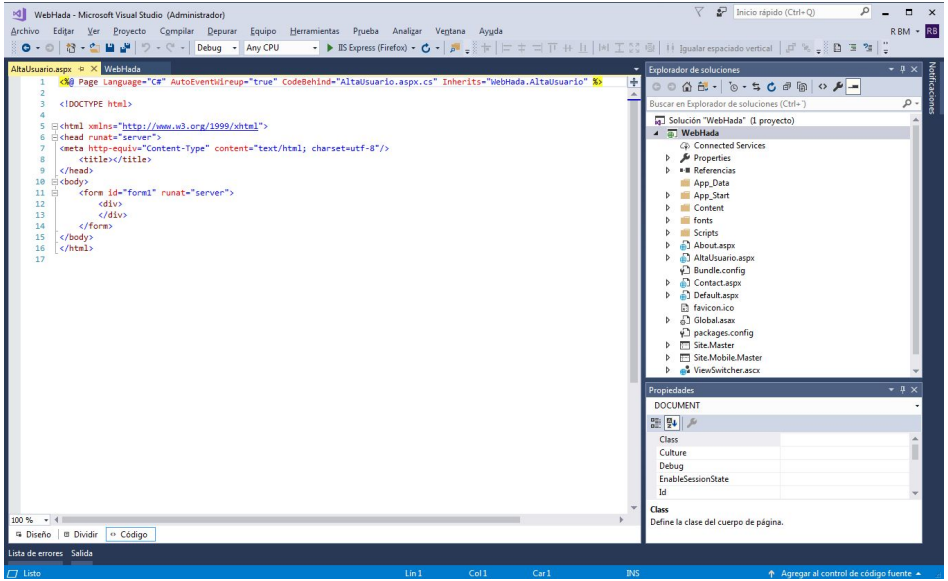
Aplicación web utilizando Web Forms

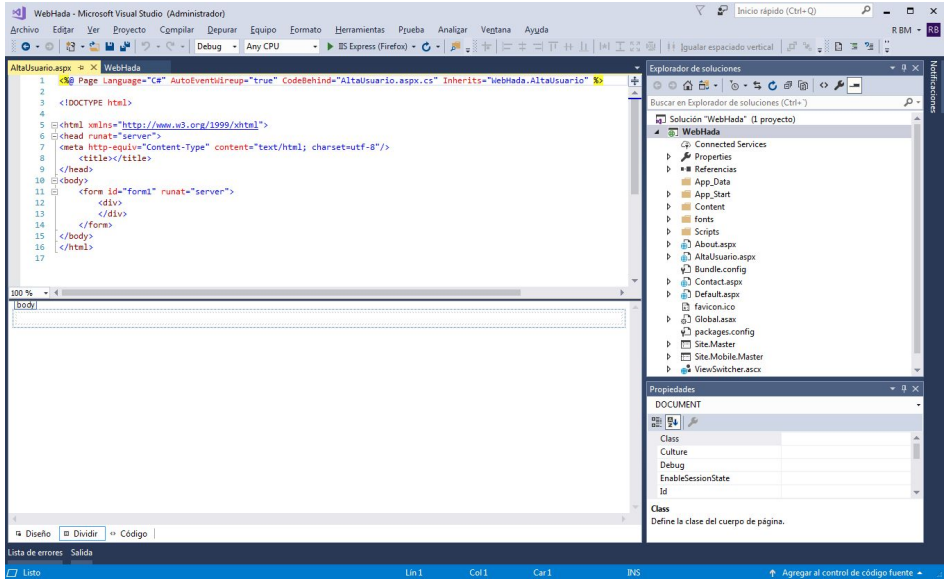
Visualización del formulario

- El IDE dispone de tres botones, que permiten cambiar el modo de visualización del código de la aplicación:
    - **Código.** Muestra el código fuente equivalente que genera la página.
    - **Diseño.** Muestra una pantalla para diseñar nuestra aplicación de manera visual. Se pueden arrastrar los componentes (botones, etiquetas, imágenes, calendarios, datos, etc.) existentes en la barra de herramientas
    - **Dividir.** Muestra una parte de la pantalla el diseño y la otra parte el código.
- Cuando añadimos algún componente desde la pantalla de *Diseño*, automáticamente se genera el código equivalente. De la misma manera, si añadimos código desde la pantalla de *Código*, se genera automáticamente su equivalente a nivel visual en la pantalla *Diseño*.

Aplicación web utilizando Web Forms

Visualización del formulario

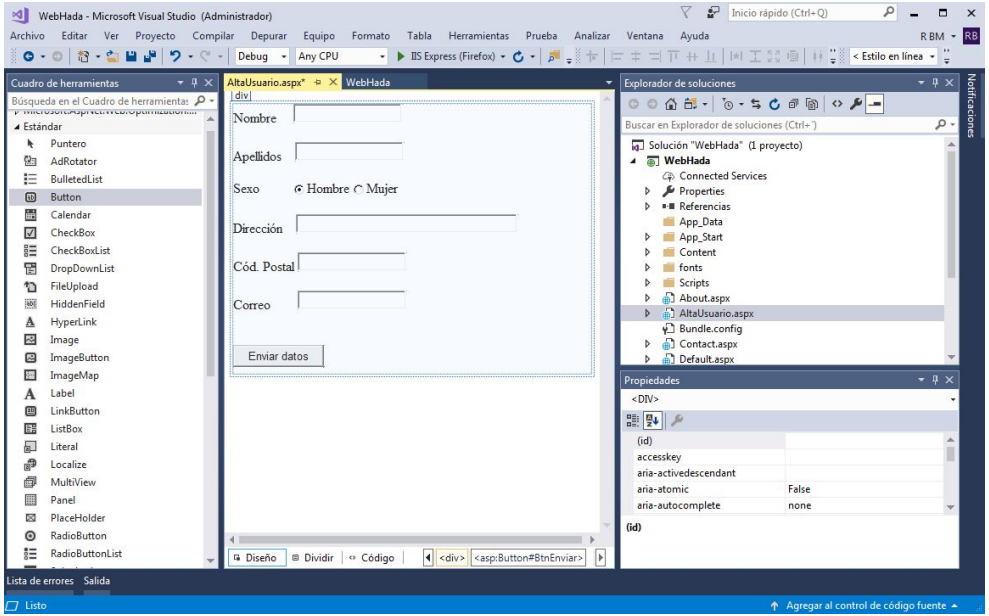




Aplicación web utilizando Web Forms

Modificar el formulario

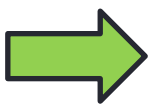
- Incorporamos los elementos arrastrando los componentes de la barra de herramientas en la vista de diseño.
- Para visualizar la barra de herramientas pulsamos el menú *Ver-Cuadro de herramientas*

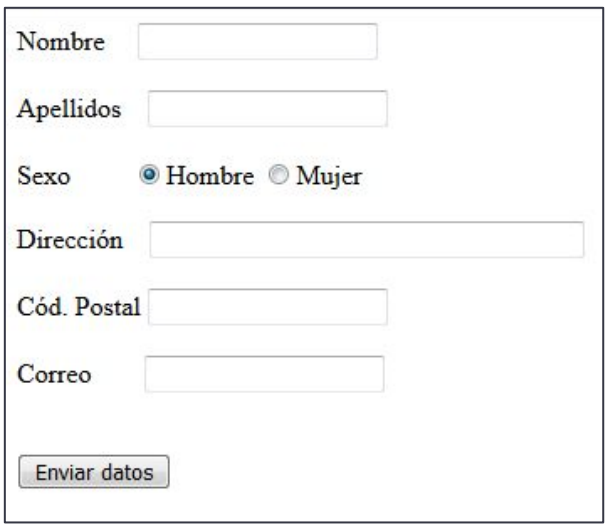


Aplicación web utilizando Web Forms

Código y ejecución

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.AltaUsuario" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div style="height: 318px">
            Nombre    
            <asp:TextBox ID="TBNombre" runat="server"></asp:TextBox>
            <br />
            <br />
            Apellidos   
            <asp:TextBox ID="TBApellidos" runat="server"></asp:TextBox>
            <br />
            <br />
            Sexo         
            <asp:RadioButton ID="RBHombre" runat="server" Checked="True" Text="Hombre" />
            <asp:RadioButton ID="RBMujer" runat="server" Text="Mujer" />
            <br />
            <br />
            Dirección    <asp:TextBox ID="TBDireccion" runat="server" Width="253px"></asp:TextBox>
            <br />
            <br />
            Cód. Postal <asp:TextBox ID="TBCodPostal" runat="server"></asp:TextBox>
            <br />
            <br />
            Correo        
            <asp:TextBox ID="TBCorreo" runat="server"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Button ID="BtnEnviar" runat="server" Text="Enviar datos" />
        </div>
    </form>
</body>
</html>
```





Controles de servidor

3

Los controles de servidor…

- Tienen propiedades que pueden ser establecidas
  - **declarativamente (en la etiqueta)**
  - **o mediante programación (en el código).**

- Junto con la página, **tienen eventos** que los desarrolladores pueden manejar
  - para ejecutar acciones especificas durante la ejecución de la página
  - o en respuesta a una acción del lado del cliente que envía la página al servidor.

Controles de servidor

- Soporte para características avanzadas: enlace de datos, plantillas..

- Se emplea el prefijo **asp:** junto con el atributo **runat="server"**.

```
<asp:TextBox id="text" runat="server"/>
```

Tipos de controles ASP.NET

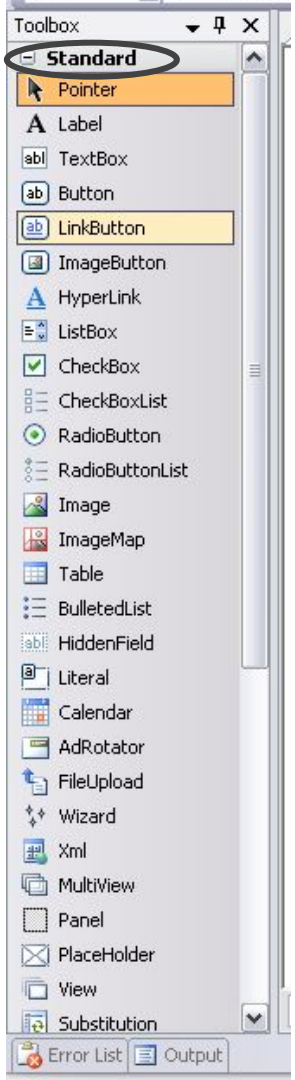
- 1. Controles HTML (no ASP.NET)
- **2. Controles estándar**
- **3. Controles de validación**
- 4. Controles de login
- 5**. Controles de navegación**
- 6. Controles Webparts
- 7. **Controles de datos**
- 8. **Controles de usuario**

Principales controles web (estándar)

Label	Se utiliza para mostrar texto dinámico (cambiamos sus propiedades a través del código del servidor)
Hyperlink	Muestra un enlace a otra página.
TextBox	Permite introducir texto al usuario. Propiedad Textmode valores: Single, Multiline o Password.
Image	Sirve para mostrar una imagen en la página web.
Button	Se usa en un Formulario web para crear un control de tipo submit (evento OnClick) .
LinkButton	Apariencia de hipervínculo pero funciones de control de un botón (envío formulario)
ImageButton	Muestra una imagen que maneja eventos tipo click.
Checkbox	Sirven para añadir casillas de verificación a una página.
RadioButton	Crea un botón de radio individual en la página.

Principales controles web

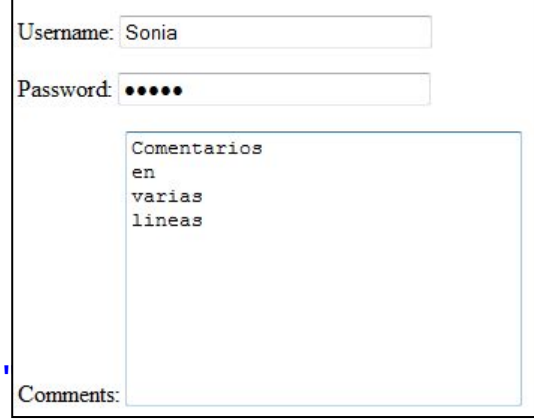
DropDownList	Proporciona un buen método para que los usuarios elijan elementos de una lista en un espacio pequeño. Cada elemento se crea con un control ListItem.
ListBox	Propiedad SelectionMode: Single, Multiple. Cada elemento se crea con un control ListItem.
CheckBoxList	Permite presentar una lista de opciones pudiendo el usuario seleccionar varias.
RadioButtonList	Permite crear una lista de botones de radio de opciones excluyentes.
Panel	Puede usarse como contenedor de otros controles.
Table, TableRow, TableCell	Permiten crear dinámicamente una tabla mediante programación.



Controles estándar

- AdRotator
- PlaceHolder
- ImageMap
- BulletedList
- HiddenField
- FileUpload
- Wizard
- Xml
- MultiView
- Substitution…

TextBox



Página aspx

```
<form id="form1" runat="server">
<div>
<p>
      Username: <asp:TextBox id="userTextBox" TextMode="SingleLine"
      Columns="30" runat="server" />
</p>
<p>
      Password: <asp:TextBox id="passwordTextBox"
      TextMode="Password" Columns="30" runat="server" />
</p>
<p>
      Comments: <asp:TextBox id="commentsTextBox"
      TextMode="MultiLine" Columns="30" Rows="10“ runat="server" />
</p>
</div>
</form>
```

Button

Página aspx

```
<form id="form1" runat="server">
<asp:Button id="BotonEnviar" Text="Enviar" runat="server“ OnClick="WriteText" />
<asp:Label id="Label1" runat="server" />
</form>
```

Página aspx.cs

```
protected void WriteText(object sender, EventArgs e)
{
        Label1.Text = "Hola mundo";
}
```

ImageButton

Página aspx

```
<form id="form1" runat="server">
   <div>
      <asp:ImageButton id="BotonImagen" ImageUrl="~/garfield.gif“ runat="server“
   OnClick="WriteText" />
         <asp:Label id="Label4" runat="server" />
      </div>
</form>
```

Página aspx.cs

```
protected void WriteText(object sender, ImageClickEventArgs e)
{
    Label4.Text = "Coordenadas:" + e.X + "," + e.Y;
}
```

Coordenadas:45,64

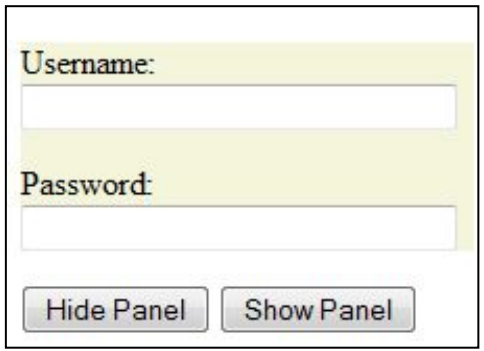
Panel

Página aspx

```
<form id="form1" runat="server">
<asp:Panel id="myPanel" BackColor="Beige" Width="220" runat="server">
<p>Username: <asp:TextBox id="usernameTextBox" Columns="30“ runat="server" /></p>
<p>Password: <asp:TextBox id="TextBox1“ TextMode="Password" Columns="30" runat="server" /></p>
</asp:Panel>
<asp:Button id="hideButton" Text="Hide Panel" OnClick="HidePanel“ runat="server" />
<asp:Button id="showButton" Text="Show Panel" OnClick="ShowPanel“ runat="server" />
</form>
```

Página aspx.cs

```
protected void HidePanel(object sender, EventArgs e)
{  myPanel.Visible = false;   }

protected void ShowPanel(object sender, EventArgs e)
{   myPanel.Visible = true;  }
```



Páginas maestras

4

Página maestra

- Una buena manera de desarrollar nuestras aplicaciones es implementar una p**ágina maestra de la que se basen el resto de páginas web de nuestra aplicación**
- Las páginas maestras permiten crear un diseño coherente para las páginas de la aplicación
- Se puede definir el aspecto, el diseño y el comportamiento estándar que desea que tengan todas las páginas (o un grupo de páginas) de la aplicación en una sola página maestra
- **A continuación, se crean páginas de contenido individuales** que incluyan el contenido que desea mostrar
- Cuando los usuarios solicitan las páginas de contenido, **éstas son combinadas con la página maestra** con el fin de generar una salida que combine el diseño de la página maestra con el de la página de contenido

Principales ventajas

- Realización de cambios de diseño en una sola ubicación; los cambios se verán reflejados en todas las páginas que usan la página maestra.

- Reutilización de la interfaz de usuario

- Mejor experiencia del usuario final: páginas más coherentes

Cómo funciona

- Una página maestra es un archivo ASP.NET con extensión *.master*, que tiene un diseño predefinido, que puede incluir texto estático, elementos HTML y controles de servidor
- **La página maestra se identifica mediante la directiva *@ Master*, que reemplaza la directiva *@ Page* que se usa en las páginas .aspx ordinarias**
- Además del texto estático y los controles que aparecerán en todas las páginas, la página maestra también incluye uno o varios controles ***ContentPlaceHolder***. Estos controles ***PlaceHolder*** definen las áreas que incluirán contenido reemplazable. A su vez, el contenido reemplazable se define en las páginas de contenido.

@Page directiva / @Master directiva

El `<%@ Page %>` directiva especifica los atributos específicos de página utilizados por el motor ASP.NET al analizar y compilar la página. Esto incluye su archivo de página maestra, la ubicación de su archivo de código y su título, entre otros datos.

```
<%@ Page Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" Title="Untitled Page" %>
```

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

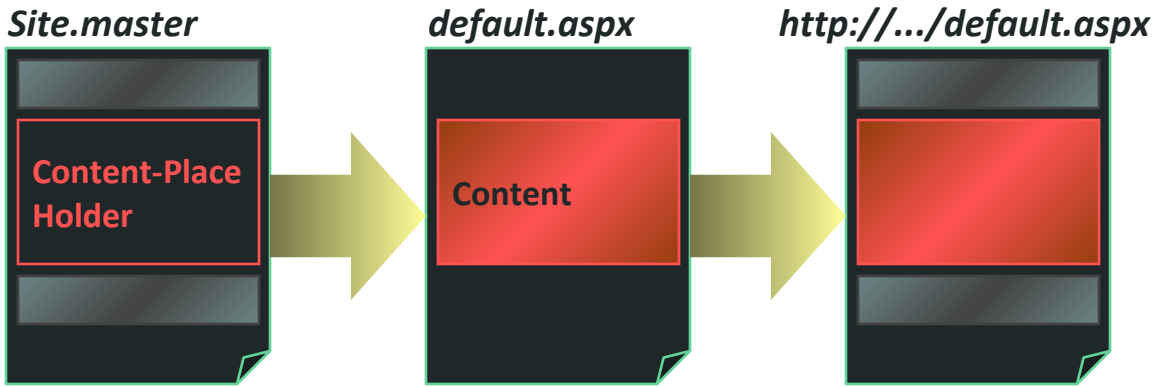
@Page directive / @Master directive

AltaUsuario.aspx | WebHada

```
1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AltaUsuario.aspx.cs" Inherits="WebHada.AltaUsuario" %>
2
3  <!DOCTYPE html>
4
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head runat="server">
7  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8      <title></title>
9  </head>
10 <body>
11     <form id="form1" runat="server">
12         <div>
13         </div>
14     </form>
15 </body>
16 </html>
17
```

Cómo funciona...

- Master pages definen el contenido común y los contenedores de contenido (content placeholders)
- Content pages hacen referencia a las paginas maestras y llenan a los contenedores con su contenido.



En tiempo de ejecución

- IIS controla las páginas maestras en la secuencia siguiente:
  - Los usuarios solicitan una página escribiendo la dirección URL de la página de contenido.
  - Cuando se captura la página, se lee la directiva @ Page. Si la directiva hace referencia a una página maestra, también se lee la página maestra. Si las páginas se solicitan por primera vez, se compilan las dos páginas.
  - La página maestra con el contenido actualizado se combina en el árbol de control de la página de contenido.
  - El contenido de los controles Content individuales se combina en el control contentplaceholder correspondiente de la página maestra.
  - La página combinada resultante se representa en el explorador.

Page.Master

- **Nueva propiedad (Master) de System.Web.UI.Page**
- Esta propiedad contiene una referencia a la página maestra de la página de contenido, por tanto provee a una página contenido de acceso programático a la página maestra
  - Determina si la pagina tiene asociada una maestra
  - Acceso a los controles definidos en la maestra pudiendo escribir código soporte en las páginas de contenido
  - Acceso a métodos públicos y propiedades definidas en la maestra
- Integración a nivel código de las páginas maestras y contenidos

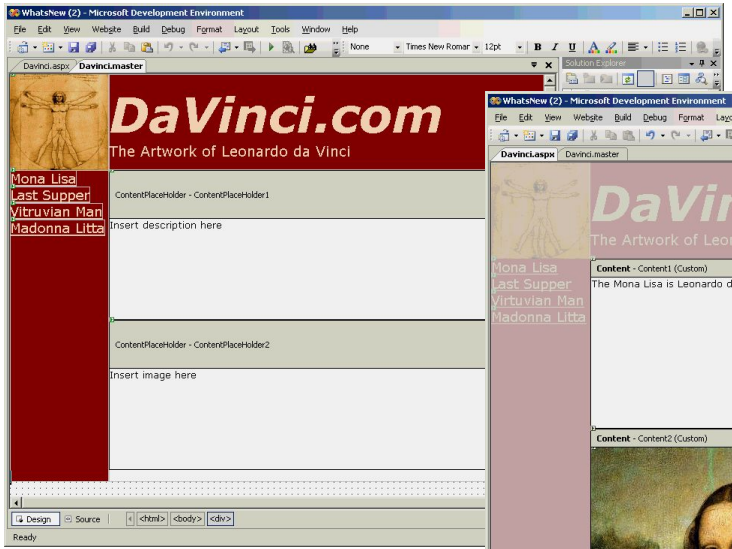
Desde el navegador..

- Desde la perspectiva del usuario, la combinación de las páginas maestras y las páginas de contenido da como resultado una única página. La dirección URL de esta página es la de la página de contenido.

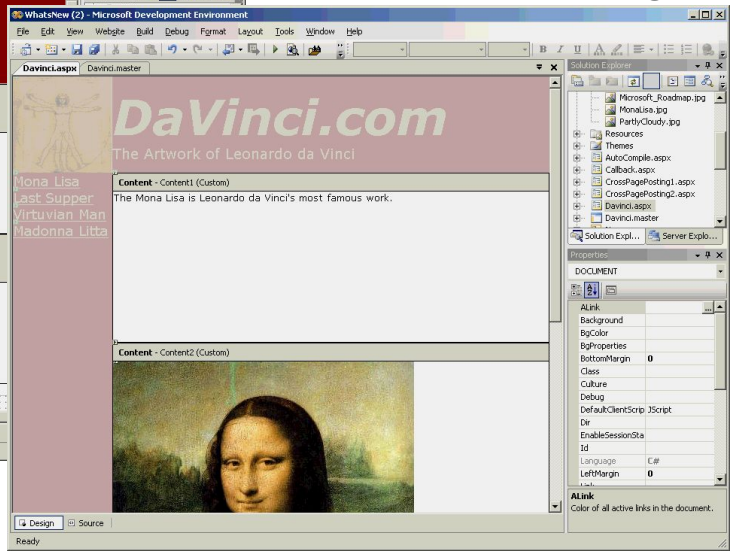
En Visual Studio…

- Herencia visual

Master Page



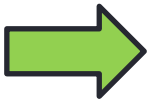
Content Page



Aplicación web (Página maestra)

- Vamos a modificar nuestra aplicación web, para que se base en una página maestra.
- Modificaremos la página maestra ***Site.Master***, que viene por defecto.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="WebHada.SiteMaster" %>

<!DOCTYPE html>

<html lang="en">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title><%: Page.Title %> - Mi aplicación ASP.NET</title>
    <asp:PlaceHolder runat="server">
        <%: Scripts.Render("~/bundles/modernizr") %>
    </asp:PlaceHolder>
    <webopt:bundlereference runat="server" path="~/Content/css" />
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
</head>
<body>
    <form runat="server">

        <div class="container">
            <h2>MI PRIMERA APLICACIÓN WEB ASP.NET </h2>

            <div class="collapse navbar-collapse">
                <ul class="nav navbar-nav">
                    <li><a runat="server" href="~/">Inicio</a></li>
                    <li><a runat="server" href="~/AltaUsuarioHija">Alta de usuario</a></li>
                    <li><a runat="server" href="~/Pedidos">Pedidos</a></li>
                    <li><a runat="server" href="~/About">Acerca de</a></li>
                    <li><a runat="server" href="~/Contact">Contacto</a></li>
                </ul>
            </div>
        </div>

        <div class="container body-content">
            <hr />
            <asp:ContentPlaceHolder ID="MainContent" runat="server"></asp:ContentPlaceHolder>
            <hr />

            <footer>
                <p>&copy; <%: DateTime.Now.Year %> - Mi primera aplicación ASP.NET</p>
            </footer>
        </div>
    </form>
</body>
</html>
```



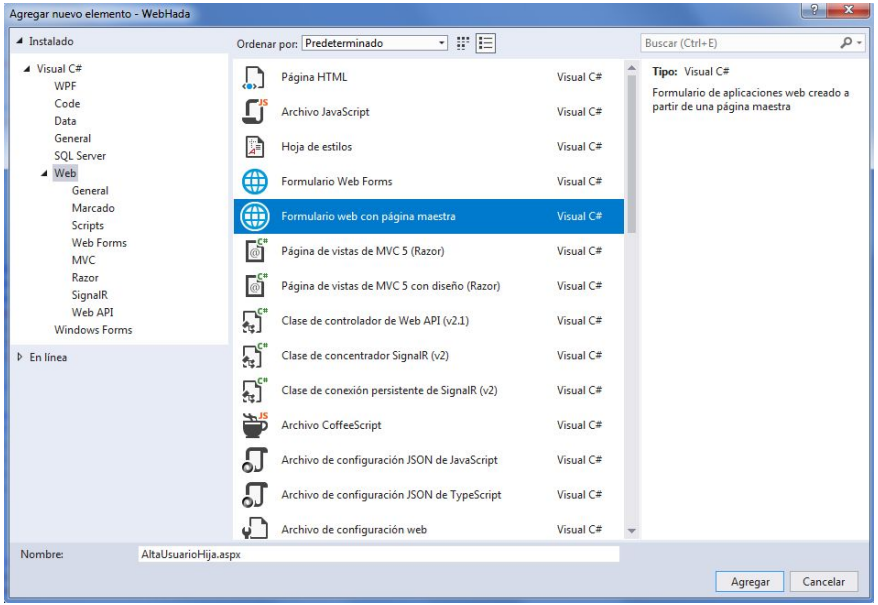
```
<%@ Master Language="C#“
AutoEventWireup="true"
CodeBehind="Site.master.cs"
Inherits="WebHada.SiteMaster" %>
```

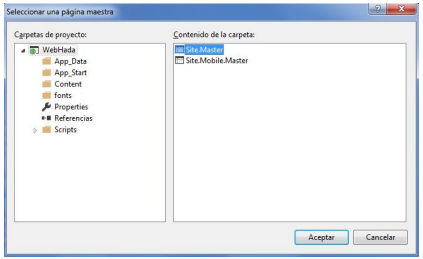
```
<asp:ContentPlaceHolder ID="MainContent"
runat="server">
</asp:ContentPlaceHolder>
```

Aplicación web (Página maestra)

Creación de formulario web con página maestra (página hija)

- Creamos un nuevo formulario con página maestra, hay que acceder al menú *Agregar -> Nuevo elemento -> Formulario web con página maestra.*
- Le damos un nombre al formulario, en este caso *AltaUsuarioHija.aspx*
- Tenemos que elegir a partir de que página maestra lo creamos





Aplicación web (Página maestra)

Contenido del nuevo formulario web con página maestra

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
CodeBehind="AltaUsuarioHija.aspx.cs" Inherits="WebHada.AltaUsuarioHija" %>

<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
</asp:Content>
```

Aplicación web (Página maestra)

Contenido del nuevo formulario web con página maestra

- El nuevo formulario *AltaUsuarioHija* ha incorporado todo lo necesario para invocar a la página maestra. Automáticamente la nueva página incorpora la directiva *MasterPageFile* e introducirá los *Content* definidos en la página maestra utilizando *ContentPlaceHolderID*.
- A las página hijas se ha añadido los siguientes elementos:
  - En la primera línea, el atributo *MasterPageFile* en el Tag Page.

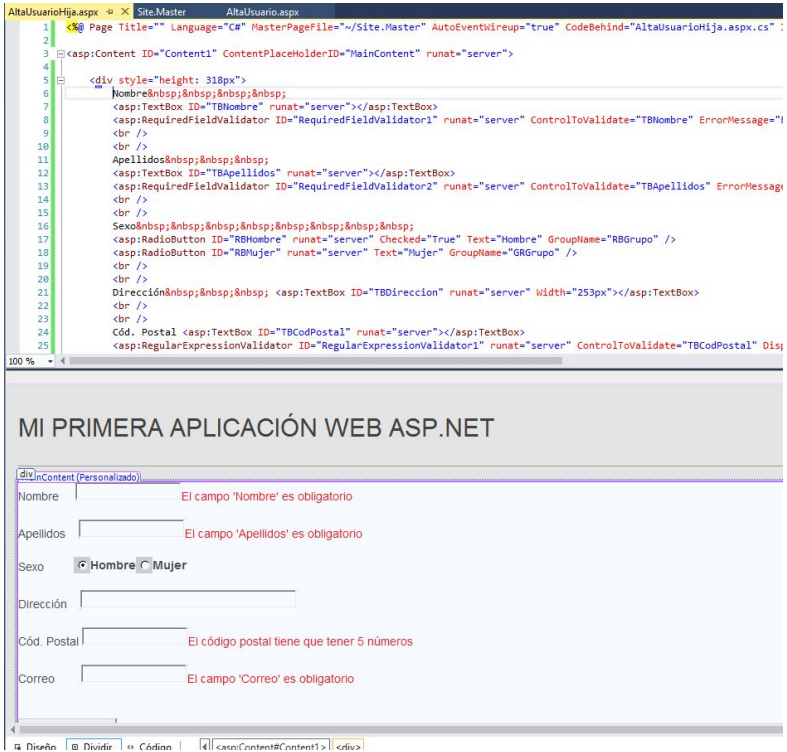
    MasterPageFile="~/Site.Master”
  - Hacer referencia al contenido mediante *Content*. La página hija, completa el contenido definido de la página maestra.

<asp:Content ID="Content1" ContentPlaceHolderID=“MainContent" Runat="Server">

Aplicación web (Página maestra)

Modificamos el nuevo formulario web

- Insertamos el código que modifique el contenido, *entre <asp:Content> y </asp:Content>*
- Copiamos el código que teníamos de la página *AltaUsuario.aspx*, y lo incrustamos
- De esta manera, cuando se llame a la nueva página, se cargará la página maestra y a continuación incorporará los contenidos



Aplicación web (Página maestra)

Ejecución

Eventos de los controles del servidor

5

Modelo de eventos

- En las páginas Web ASP.NET, los eventos asociados a los controles de servidor se originan en el cliente (explorador) pero los controla la página ASP.NET en el servidor Web.

- La información del evento se captura en el cliente y se transmite un mensaje de evento al servidor mediante un envío HTTP.

- La página debe interpretar el envío para determinar el evento ocurrido y, a continuación, llamar al método apropiado del código del servidor para controlar dicho evento.

Enlazar eventos a métodos

- Un evento es un mensaje que envía un objeto cuando ocurre una acción (Ej. "se ha hecho clic en un botón”) . La acción puede estar causada por la interacción del usuario, como un clic, o por otra lógica del programa.

- En una aplicación, se debe traducir el mensaje en una llamada a un método del código.

- El enlace entre el mensaje del evento y un método específico (es decir, un controlador de evento) se lleva a cabo utilizando **un delegado de eventos.**

Eventos y delegados

- El objeto que provoca el evento se conoce como remitente del evento. El objeto que captura el evento y responde a él se denomina receptor del evento.
- En las comunicaciones de eventos, el remitente del evento no sabe qué objeto o método recibirá los eventos que provoca. **Se necesita un intermediario (o mecanismo de tipo puntero) entre el origen y el receptor.**
- .NET Framework define un tipo especial (**Delegate**) que proporciona la funcionalidad de un puntero a función.

Manejadores de eventos

- El prototipo de los métodos manejadores de eventos deben coincidir con el prototipo del *delegado EventHandler*.

```
delegate void EventHandler(object sender, EventArgs e);
```

- Por tanto, los manejadores de evento en ASPnet devuelven ***void*** y tienen dos parámetros:
  - ***Objeto que lanza el evento:*** objeto (*Object sender*)
  - ***Argumentos del evento*:** información específica del evento (*EventArgs* o tipo derivado)

  - Por ejemplo, para un control ImageButton de servidor Web, el segundo argumento es de tipo ImageClickEventArgs, que incluye información sobre las coordenadas donde el usuario ha hecho clic..

Asociar el manejador al control

- Escribirlo manualmente

Archivo Default.aspx

```
<asp:Button ID="Button1" runat="server" onclick="EventoClick"
    Text="Button" />
```

Archivo Default.aspx.cs

```
protected void EventoClick(object sender, EventArgs e)
      { }
```

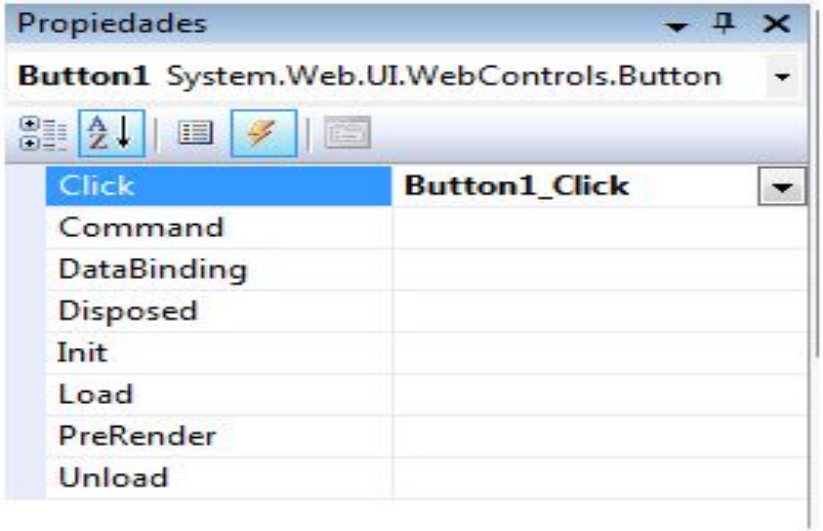
- Pulsar doble click sobre un control en vista diseño (evento por defecto)

```
protected void Button1_Click(object sender, EventArgs e)
```

Al compilar la página, ASP.NET busca un método denominado *EventoClick* y confirma que éste tiene la firma adecuada (acepta dos argumentos, uno de tipo **Object** y otro de tipo **EventArgs**). A continuación, ASP.NET enlaza automáticamente el evento al método

Asociar el manejador al control

- Seleccionar el evento de la ventana de propiedades del control (y asociar un nombre de método manejador del evento o doble click)



Eventos postback vs no-postback

- Una página se carga después de cada petición: fenómeno conocido como **PostBack (=envío).**

- Eventos **Postback**:
  - causan que la información del formulario se envíe al servidor inmediatamente.

- Eventos **no-Postback (o cached)**:
  - la información se envía en el siguiente evento **postback**.
  - Los eventos se guardan en una cola en el cliente hasta que un evento **postback** ocurre.

Eventos postback vs no-postback

- Button, Link Button y Image Button causan eventos **postback.**

- TextBox, DropDownList, ListBox, RadioButton y CheckBox, proveen eventos **cached.**
  - Sin embargo podemos sobrecargar este comportamiento en los controles para poder realizar eventos postback, cambiando la propiedad ***AutoPostback*** a true**.**

Eventos de página

- Las páginas ASP.NET provocan eventos de ciclos de vida como **Init**, **Load**, **PreRender** y otros.
- De manera predeterminada, los eventos de página se pueden enlazar a los métodos utilizando la convención de nomenclatura **Page_***nombreDeEvento*.
  - Por ejemplo, con el fin de crear un controlador para el evento **Load** de la página, se puede crear un método denominado **Page_Load**.
  - En tiempo de compilación, ASP.NET buscará los métodos que se basen en esta convención de nomenclatura y realizará el enlace automáticamente entre el evento y el método.
- Se puede utilizar la convención **Page_***nombreDeEvento* para cualquier evento expuesto por la clase **Page**.

Propiedad IsPostBack

- Una página se carga después de cada petición: fenómeno conocido como ***PostBack*** (=envío)
- El evento ***Page_Load*** se produce en cada petición
- ***Page.IsPostBack*** *para ejecutar código condicional*

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack) {
        Response.Write("<br>Page has been posted back.");
    }
}
```

Importante: Los métodos de control de eventos de página no requieren ningún argumento. Estos eventos (como Load) pueden aceptar los dos argumentos estándar, pero en estos argumentos no se pasa ningún valor.

Navegación entre formularios

6

Navegación tradicional

- En HTML:
  - Elemento <a> y atributo href

```
<a href=”Login.aspx”>Regístrate aquí </a>
```

  - La página Login.aspx debe estar en el mismo directorio que la página que contiene el enlace
  - Después de pulsar sobre el enlace se muestra la página Login.aspx
- En ASPx:

```
<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/Login.aspx">Regístrate aquí</asp:HyperLink>
```

Diferentes formas de navegar

- **Control hipervínculo**
  - Navega a otra página

- Método **Response.Redirect**
  - Navega a otra página por medio del código. (Equivalente a navegar a través de un enlace)

Hipervínculos y Redirección

- Los hipervínculos responden a eventos click mostrando la página especificada en la propiedad *NavigateURL* del control.

- Si quieres capturar un click en el código, debes usar los eventos de un *LinkButton* o *ImageButton* y utilizar

```
Response.Redirect("SiguientePagina.aspx");
```

- También podemos pasar parámetros a la nueva página.

Paso de parámetros a otra página

- Paso de parámetros en la URL.
  - Almacena los datos en la colección QueryString
  - Múltiples parámetros separados por &

```
int valor = 22;
int valor1 = 25;
Response.Redirect("Default4.aspx?par1=" + valor);
Response.Redirect("Default4.aspx?par1=" + valor + "&par2=" + valor1);
```

http://localhost:49999/Default4.aspx?par1=22&par2=25

Paso de parámetros a otra página

- Limitaciones con QueryString
  - Los caracteres utilizados deben ser caracteres permitidos en una URL
  - La información es visible a los usuarios en la barra del navegador
  - Los usuarios pueden modificar la información provocando errores inesperados
  - Muchos navegadores imponen un límite en la longitud de la URL

Paso de parámetros a otra página

- Caracteres especiales:
  - & (para separar múltiple query strings)
  - + (alternativa para representar un espacio)
  - # (especifica un marcador en una página web)
- Solución:
  - Utilizar los métodos de la clase *HttpServerUtility* para codificar los datos

```
Response.Redirect("WebForm2.aspx?par1="+ Server.UrlEncode(" &hola "));
```

Paso de parámetros a otra página

- El método *UrlEncode* reemplaza los caracteres especiales por secuencias de escape

  http://localhost:49999/WebForm2.aspx?par1=+%26hola+

- La recuperación de datos codificados mediante el método UrlEncode con QueryString es automática en ASP.NET (no es necesario utilizar un método que decodifique los datos)

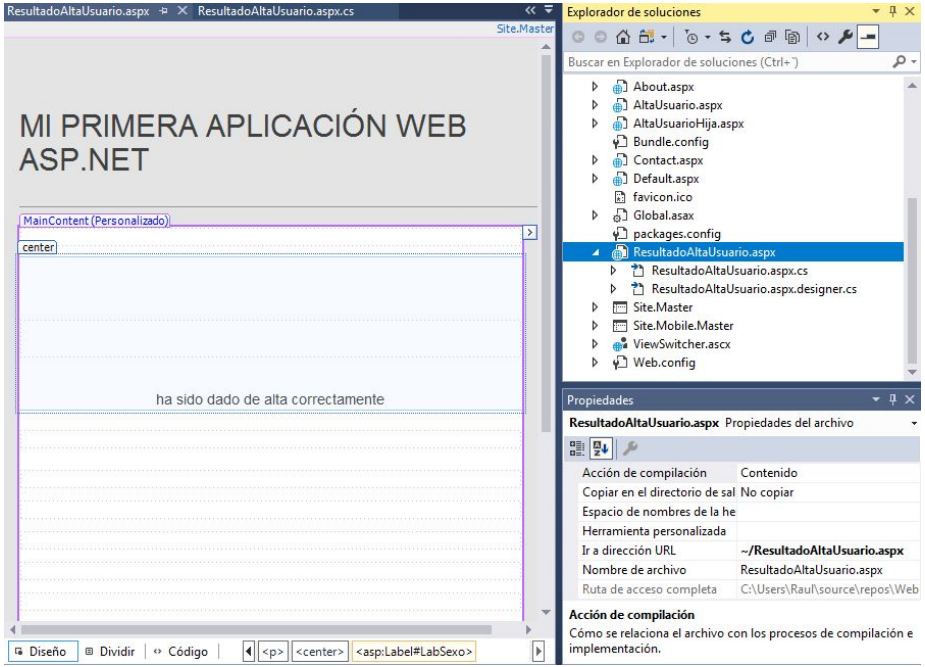
Objeto Request

- Proporciona información sobre la petición HTTP del cliente que ha provocado la carga de la página actual.
  - Información sobre el cliente (Request.Browser, Request.Browser.IsMobileDevice, Request.Browser.Id)
  - Cookies (Request.Browser.Cookies)
  - Parámetros pasados a la página:

```
if (Request.QueryString["par1"] != "")
    Label1.Text = Request.QueryString["par1"];
```

Aplicación web (Navegación)

- Vamos a modificar nuestra aplicación web, para que una vez introducidos los datos, y pulsemos en el botón *Enviar datos,* nos redirija a otra página de resultado al que le pasaremos el nombre, los apellidos y el sexo como parámetros.
- Creamos otro formulario web a partir de la página maestra *Site.master* con el nombre *ResultadoAltaUsuario.aspx*. Esta página recibirá los parámetros pasados por la página *AltaUsuarioHija.aspx*
- La página *AltaUsuarioHija.aspx*, mostrará según el sexo del usuario, la palabra “La usuaria” o “El usuario”.



Aplicación web (Navegación)

- Modificamos la página *AltaUsuarioHija.aspx*, para incorporar el evento *BtnEnviar_Click* , asociado al botón *Enviar datos.*

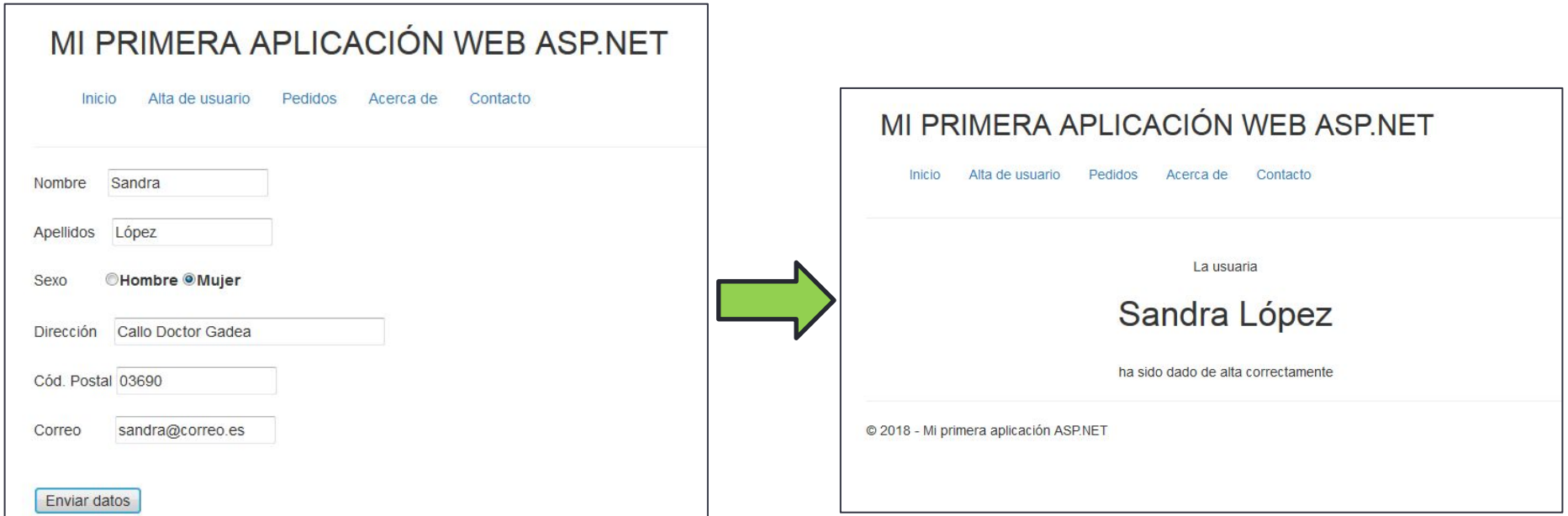
```
protected void BtnEnviar_Click(object sender, EventArgs e)
{
  String s;
  if(RBHombre.Checked)
      s=RBHombre.Text;
  else
      s=RBMujer.Text;
  Response.Redirect("ResultadoAltaUsuario.aspx?nombre="+ TBNombre.Text + "&apellidos=" + TBApellidos.Text &sexo="+s);
}
```

- Modificamos la página *ResultadoAltaUsuarioHija.aspx*, para incorporar en el evento *Page_Load,* el código para cargar los datos recibidos en una etiqueta.

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Request.QueryString["sexo"] == "Hombre")
       LabSexo.Text = "El usuario";
   else
       LabSexo.Text = "La usuaria";

   LabNombreUsuario.Text = Request.QueryString["nombre"] + ' ' + Request.QueryString["apellidos"] ;
}
```

Aplicación web (Navegación)

Ejecución



CSS

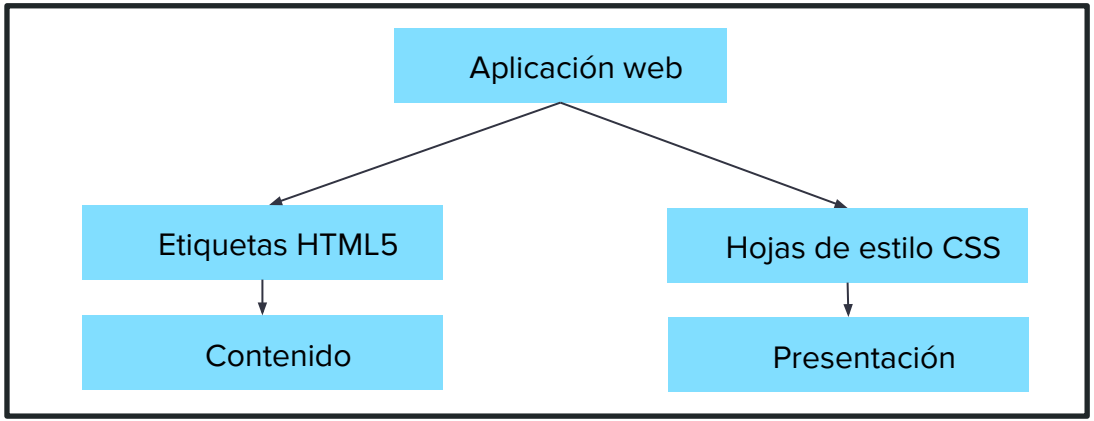
- HTML
- ASP . NET
- **CSS**

CSS

- CSS es una herramienta que permite definir la presentación de un documento
- Permite crear un conjunto de estilos en una única localización
- Las páginas a las que se aplica la misma hoja de estilos tendrán las mismas fuentes, colores y tamaño
- Proporcionan una estética homogénea en todas las páginas de un sitio web
- Las hojas de estilo son elementos agregados al lenguaje HTML que tomarán en cuenta la presentación del documento o de la aplicación web
- El mismo contenido podrá visualizarse según la hoja de estilos adoptada, de forma diferente en distintos dispositivos (PC, Tablet, móvil, etc.)

CSS

- Separación entre contenido y presentación



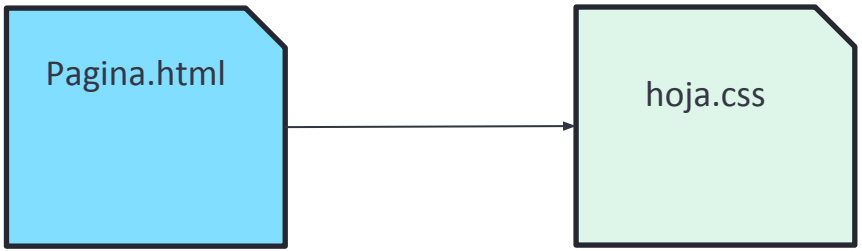
- Podemos incorporar el estilo de tres formas:
  - Una hoja de estilos externa, en un archivo distinto con extensión .css
  - Una hoja de estilos interna incrustada en el propio documento HTML
  - Una hoja de estilos en línea para aplicar a un elemento determinado

Tipos de hojas de estilo

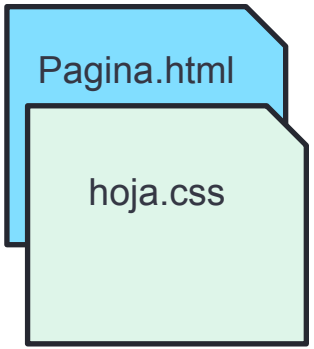
Hoja de estilo externa

- Se emplazan todas las reglas de estilo en una única hoja de estilos externa
- Se enlaza esta hoja de estilos externa con todas las páginas que vayan a tener la misma apariencia
- Para hacer una referencia a una hoja de estilos externa en una web, hay que introducir en el elemento **HEAD**, la siguiente sentencia:

```
<link rel=“Stylesheet" type="text/css" href=“~/hoja.css" />
```



Tipos de hojas de estilo (2)



Hoja de estilo interna

- Hoja de estilo incrustada dentro de un documento.
- Se utiliza las etiqueta <style>, dentro del elemento **HEAD**
- Se debe reescribir en cada página (lo cual es un problema)
- Se definen las reglas de estilo entre las etiquetas:

```
<style type="text/css">
        a { background-color: #ff9;
         color: #00f; }
</style>
```

Tipos de hojas de estilo (3)

Hoja de estilo en línea

- Permiten asignar un estilo a un elemento determinado
- Para ello hay que utilizar el atributo style dentro del elemento al que se quiere aplicar el estilo

```
<a href="/" style="background-color: #ff9;
color: #00f;"> Home</a>
```

```
<h1 style="color:blue;margin-left:60 px;"> Estilo
   en línea </h1>
```

Selector de estilos

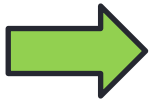
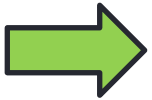
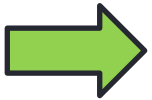
- En caso de utilizar hojas externas o internas cada regla de estilo tiene un selector
- Un selector es el tipo de elemento al que se va a aplicar el estilo
- Ejemplo:

```
a  {
    background-color: #ff9;
    color: #00f;
   }
```

- En ASP.NET hay dos tipos de selectores:
  - Tipos de elementos y clases

Selector de tipo elemento

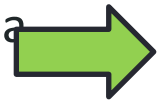
- El estilo se aplica a todos los elementos del mismo tipo

h2 { color: #369;}		Cambia de color todas las cabeceras de segundo nivel
table {text-align: center;}		Alinear el texto de las celdas de todas las tablas
p{ font-size: 150%;}		Cambia el tamaño de la fuente para todo el párrafo

Selector de tipo clase

- Se utiliza cuando asignamos a cada elemento una clase determinada

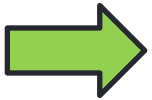
- La página HTML indicará la referencia a la clase definida en el estilo



```
<p class="pageinfo">
    Copyright 2019
</p>
```

- La hoja de estilos contendrá para cada clase una serie de características

- Los selectores de clase van precedidos por un .



```
.pageinfo
 {
   font-family: Arial;
   font-size: x-small;
 }
```

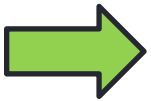
Selector de tipo clase (2)

- Otra forma de definir una clase:

```
h1.textorojo
{
    color: red;
}
```

- La invocación desde HTML será

```
<h1 class="textorojo"> Texto en rojo</h1>
<h1> Texto por defecto</h1>
```



Texto en rojo

Texto por defecto

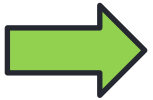
Selector de identificador

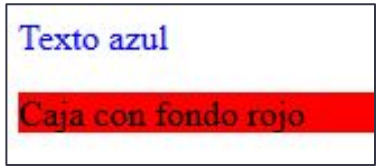
- El selector id, permite aplicar una hoja de estilo, aunque sólo se podrá invocar una vez en el documento
- Identifica un elemento único en la página

```
#textoazul {color: blue;}
#fondorojo {
    background-color: red;
}
```

- La invocación desde HTML será

```
<p id="textoazul"> Texto azul </p>
<div id="fondorojo"> Caja con fondo rojo </div>
```





Ejemplo de hoja de estilos css

```
html,body {
   height:100%;
}

h1 {text-align: center;}

header {
   display:block;
   background:#086af0;
   padding:6px 10px;
   color:white;
}
section {
   width: 79%;
   background: #ccc;
   float: left;
   overflow: auto;
   padding-bottom: 60px;
   padding-top:30px;
}

aside {
   float: right;
   border: 1px solid red;
   width: 19%;
   border: 1px solid red;
}
```

```
footer {
   position: relative;
   margin-top: -50px;
   height: 40px;
   padding:5px 0px;
   clear: both;
   background: #286af0;
   text-align: center;
   color: white;
}
figure {
   display: table; margin: 0 auto;
}
nav {
    position : absolute;
    width : 100%;
    height: 50px;
    background-color: #333;
    overflow: hidden;
    color: white;
}
nav ul {
    margin : 0 auto;
    width : 940px;
    list-style : none;
}
```

```
nav ul li {
    float : left;
}

nav ul li a {
    display : block;
    margin-right : 20px;
    width : 140px;
    font-size : 18px;
    line-height : 44px;
    text-align : center;
    text-decoration : none;
    color : white;
}

nav ul li a:hover {
    color : #fff;
}
nav ul li.selected a {
    color : #fff;
}
```

Propiedades de estilo

- Font: Tipo de fuente, tamaño, color, negrita…
- Background: color de fondo, imagen de fondo…
- Block: espacio entre párrafos, líneas, palabras…
- Box: personalizar tablas, colores, bordes…
- Border: dibuja bordes alrededor de diferentes elementos
- …

CssClass

- Asociar selector de tipo clase en Formularios Web en ASP.NET:

```
<head runat="server">
<title>Probando CSS</title>
<link rel="Stylesheet" type="text/css" href="hoja.css" />
</head>
```

hoja.css

```
.textbox
{ font-family: Arial;
background-color:0099FF;
border: 1px solid
}
```

```
<asp:TextBox ID="TextBox1" CssClass="textbox" runat="server" />
```

Maquetación con CSS

- ***maquetar una pagina web es pasar el diseño a código HTML***, poniendo cada cosa en su lugar (una cabecera, un menu, etc.).
- Hasta hace unos años la única manera de maquetar una pagina web era mediante tablas HTML (<table>), pero esto tiene muchas desventajas y limitaciones
  - el uso de las tablas está condicionado a la mera tabulación de datos,
  - Un diseño con tablas no es flexible, es decir, que no podemos cambiar la distribución de los elementos en la página, a menos que la volvamos a hacer.
  - Cada Explorador renderiza de manera distinta cada documento HTML y con estructuras con tablas el cambio es más notorio
  - Ocupa más espacio y más ancho de banda.
  - Google no indexa de igual manera las páginas con estructuras basadas en tablas.
- Por eso la técnica de maquetación fue evolucionando con los años hasta llegar al punto donde no se usan tablas, si no capas (los famosos DIVs) **a las que se le dan formato mediante CSS.**

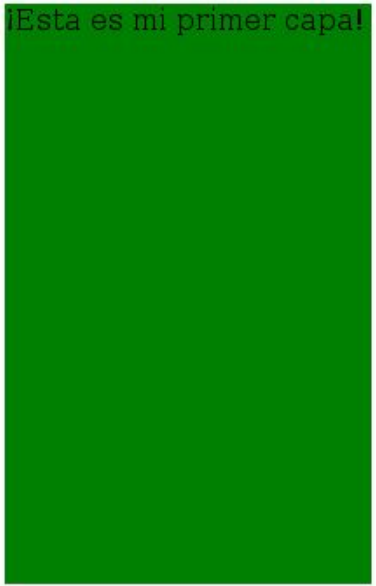
Divs

- Las capas, layouts o divs son la misma cosa con distinto nombre.
- Para tener un concepto mental de lo que son, podemos imaginarlos como ***contenedores o bloques donde podemos meter lo que queramos dentro*** *(imágenes, texto, animaciones, otro bloque, o todo al mismo tiempo) a los que se le asigna un ancho, alto y posición*, de esta manera se van a ir posicionando consiguiendo la estructura que queremos.

Formato a un DIV

- Para darle formato a un DIV tenemos que identificarlo de alguna forma, para esto existe **el atributo ID**, en el pondremos el nombre del DIV para luego llamarlo desde la hoja de estilos, la forma de escribirlo es así:

```
<div id="capa1">¡Esta es mi primer capa!</div>


#capa1{
    width:210px;
    height:300px;
    background-color:green;
}
```



Class o id?

- La diferencia entre una clase (.) y un bloque (#) es que
  - El bloque es único e irrepetible en la pagina, es decir, si creamos un estilo de bloque "**#busqueda**" para mostrar el cuadro de búsqueda no podremos usarlo otra vez en la misma pagina,
  - En cambio si a "**#busqueda**" lo convertimos en una clase "**.busqueda**" podremos usarlo cuantas veces queramos.

Propiedades de maquetación

```
<div style = "display: table; margin-left: auto; margin-right: auto; width: 500px;">
    <div style = "height: 45px; width: 500px;"></div>
    <div style = "float: left; height: 75px; width: 150px;"></div>
    <div style = "float: right; height: 75px; width: 350px;"></div>
    <div style = "clear: both; height: 35px; width: 500px;"></div>
</div>
```

- Los elementos DIV pueden centrarse utilizando los atributos **margin-left: auto; margin-right: auto;**
- **La propiedad float** ajusta los elementos hacia el margen indicado (cuando tenemos capas adyacentes).
- A veces necesitarás tener una capa o bloque que no tenga capas a su/s lados, para eso esta la propiedad CSS ***Clear***. (ej el pie de página).
  Esta propiedad se utiliza en conjunto con *float* y **sirve para evitar que una capa se posicione a cualquiera de los lados**

Cuidado

- En las estructuras clásicas con tablas, podíamos utilizar tamaños en porcentaje. Aunque aquí, también podemos utilizar porcentajes, el dibujado de la página puede no verse como se esperaba.

Maquetar con CSS links

- https://www.youtube.com/watch?v=sOyA84uhoCQ
- http://www.1keydata.com/css-tutorial/
- https://www.youtube.com/watch?v=nTOAXkMbd_0
- http://www.tutorialmonsters.com/creando-una-web-ejemplo-con-div-y-css-primera-parte/