

Departament de Llenguatges i Sistemes Informàtics

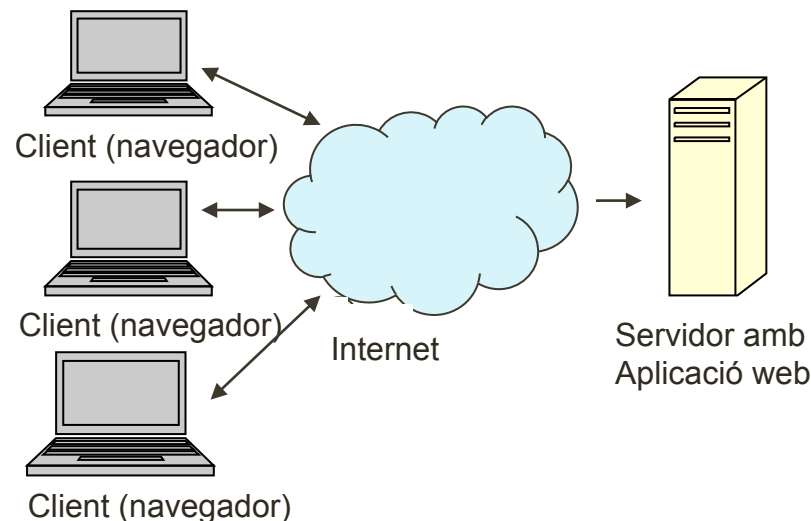
# Tema 9. Aplicacions Web. Capa d'Interfície (I)

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Escola Politècnica Superior  
Universitat d'Alacant

# Aplicacions web vs escriptori

- Simple creació d'interfície d'usuari.
- Distribució d'actualitzacions més fàcil i ràpid i menys costos.
- Processament distribuït.
  - Moltíssim més fàcil proveir processament del costat del servidor.
  - La web proveeix protocols estàndard (HTTP, HTML, XML) per facilitar aplicacions n-capa.



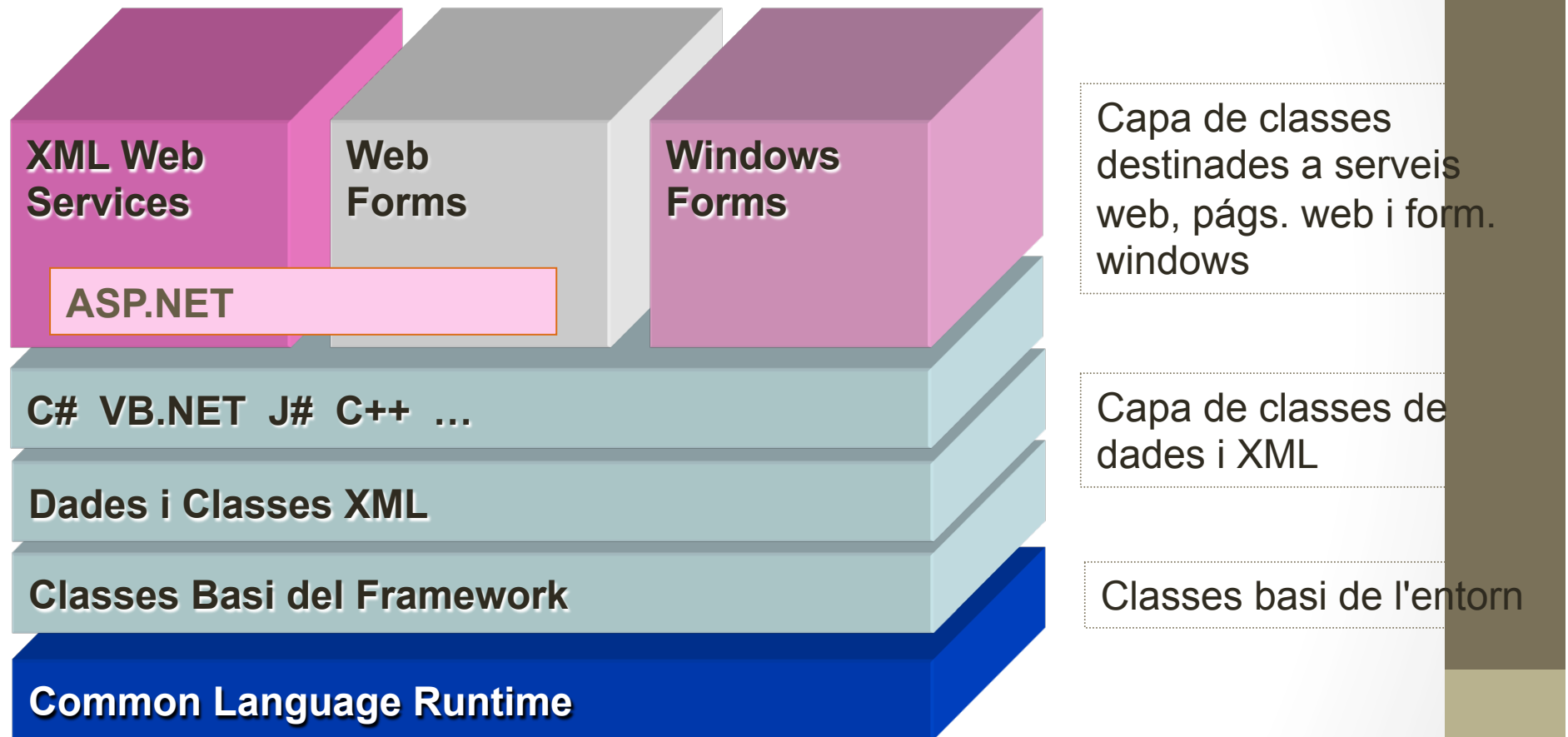
# Tecnologia depenent del servidor

- L'avantatge principal radica en **la seguretat que té el programador sobre el seu codi**:
  - Aquest es troba únicament en els arxius del servidor que en ser sol·licitat a través del web, és executat.
  - Els usuaris no tenen accés més que a la pàgina resultant en el seu navegador

# Què és ASP.NET?

- Plataforma de per construir **Aplicacions Web i Serveis Web** que funcionen sota IIS.
- Part del Framework .NET
- Tecnologia del costat del servidor
- Llenguatges ASP.NET
  - Orientats a Objecte
  - Dirigits per Esdeveniments
  - Compilats en el Servidor
- Suport de múltiples llenguatges:
  - C#
  - VB.NET
  - Jscript.NET
  - J#

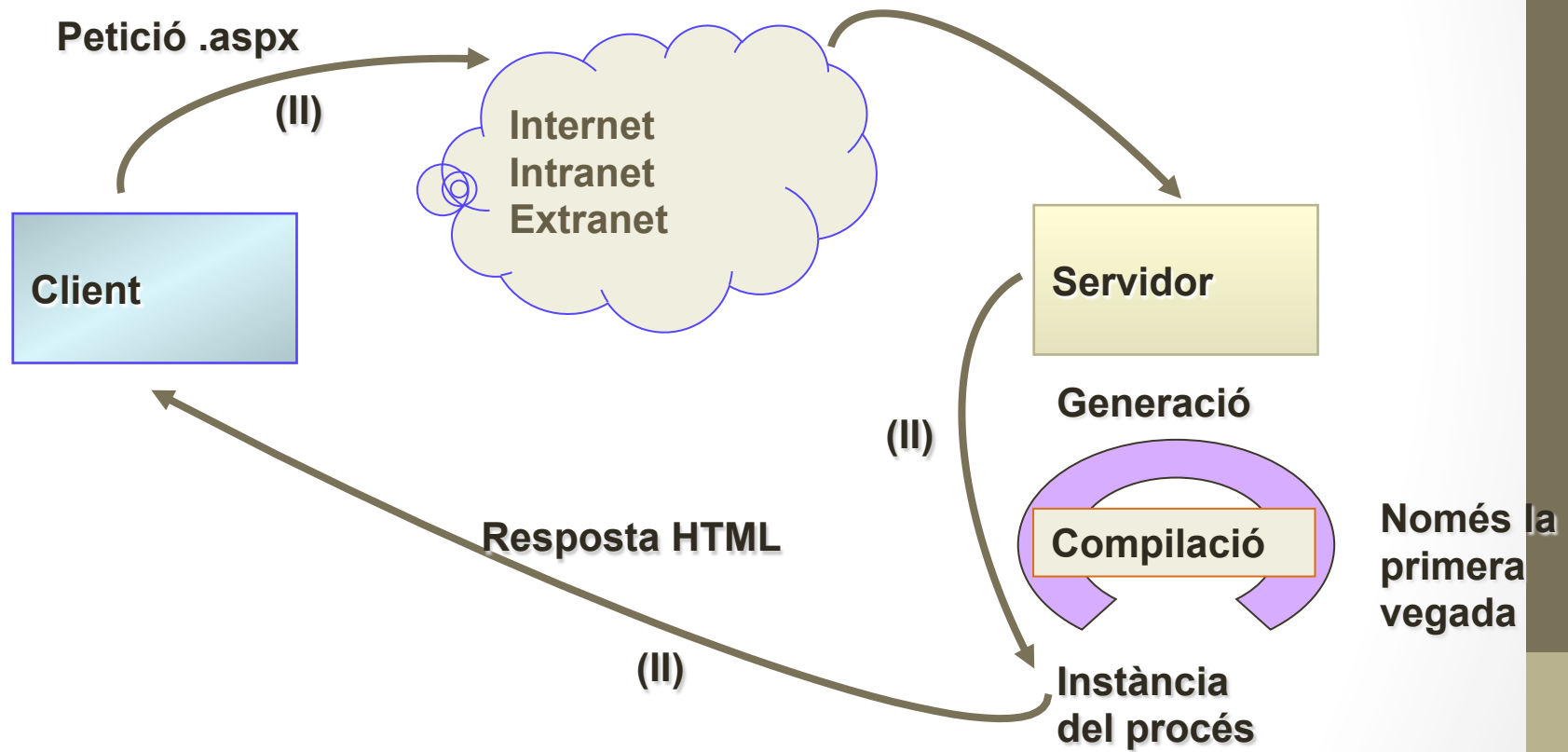
# Framework de Microsoft .NET



# “Execució”

Com funciona?

- Cridada a una pàgina ASP .NET



# Aplicacions Web ASP.NET

- Combinació d'arxius, pàgines, manejadores, mòduls i codi executable que pot invoqués des d'un **directori virtual**.
- Es divideixen a diverses pàgines web.
- Comparteixen un conjunt de recursos i opcions de configuració comuna.
- Cada aplicació té el seu propi:
  - Conjunt de caché
  - Dades d'estat de sessió

# On es guarda l'aplicació??

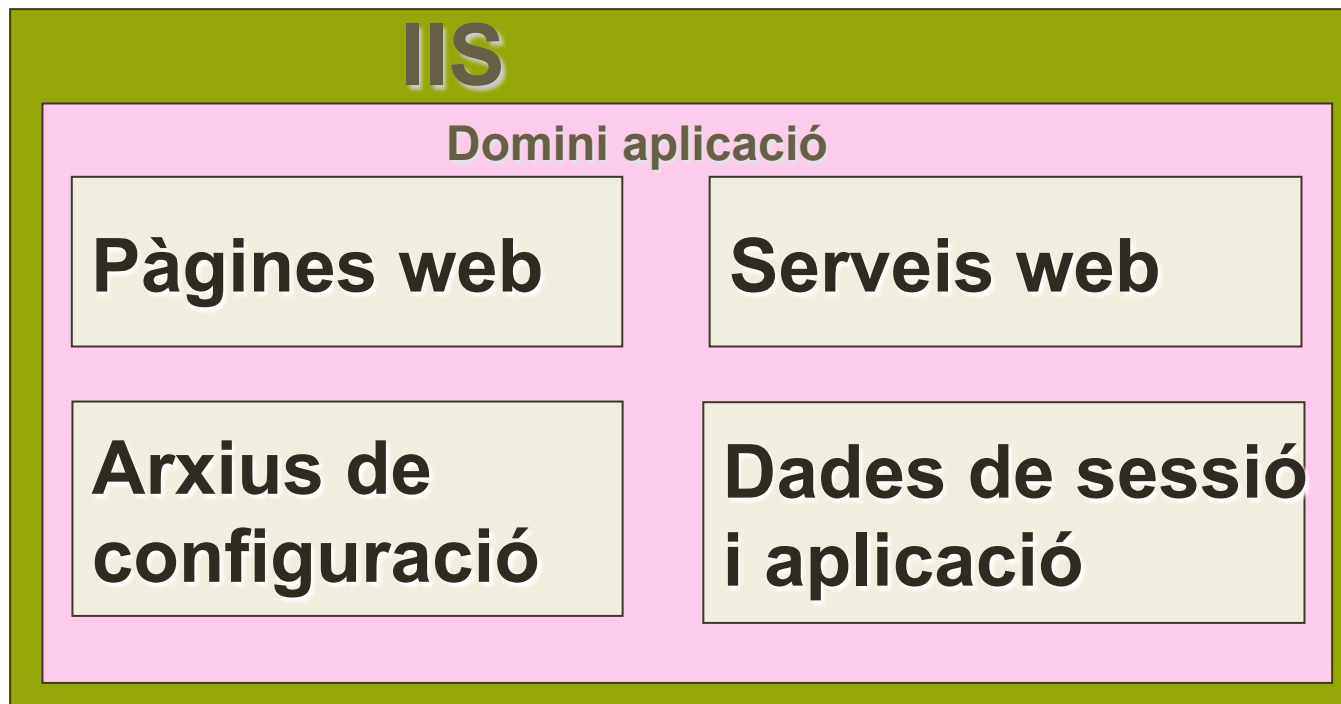
## → Directori Virtual

- Una aplicació web només existeix en una localització que ha estat publicada per IIS com un **Directori Virtual**.
- Un **directori virtual** és un recurs compartit identificat per un àlies que representa la localització física en el servidor.
- `//localhost` és la carpeta virtual arrel de l'ordinador (`\InetPub\wwwroot`)



# Directori virtual

- Directori virtual: estructura d'agrupació bàsica que delimita una aplicació.
- Creació i administració des de IIS (Internet Information Server)



# Formularis Web (I)

- Tècniques per a Ràpid Desenvolupament d'Aplicacions (RAD).
- Podem:
  - Arrossegar i deixar anar controls en un formulari
  - Escriure el codi suport
- L'aplicació es desenvolupa per a un servidor web.
- Els usuaris interactuen amb l'aplicació a través d'un navegador.

# Formularis Web (II)

- Proporcionen una aproximació orientada a:
  - objectes
  - esdeveniments
  - gestió d'estat
- Programació del costat del servidor per manejar esdeveniments del costat del client:
  - pueden executar-se, virtualment, sobre qualsevol navegador compatible amb HTML.

## Formularis Web (III)

- Tots els controls de servidor han d'aparèixer dins d'una etiqueta `<form>`, i aquesta etiqueta ha de contenir l'atribut `runat="server"`.
- Aquest atribut indica que el formulari s'ha de processar en el servidor.
- També indica que els controls que conté poden ser accedits per scripts del servidor:  
`<form runat="server">`  
    ...HTML + controls de servidor  
`</form>`
- **Una pàgina .aspx ha de contenir un únic control `<form runat="server">`**

# En VisualStudio...

## Assetjo Web o Projecto Web?

- Assetjo Web: conjunt de pàgines Web independents.
  - Para pàgines Web senzilles (ej, pàgina Web personal...)
- **Projecto Web:** conjunt de pàgines Web enllaçades amb un arxiu de projecte.
  - Para aplicacions avançades
  - Podem referenciar DLLs, etc

# Necessitem Internet Information Server?

- Visual Studio disposa del seu propi servidor de desenvolupament, per la qual cosa per fer proves en el nostre ordinador no necessitem tenir instal·lat IIS.
- No obstant això, per poder desplegar la nostra aplicació en un servidor, si ho necessitaríem.

# En Visual Studio...

## On es guarda l'aplicació?

- **File system**
- C:\Documents and Settings\aaaa\Mis documentos\Visual Studio 2005\WebSites\WebSite2
  - <http://localhost:3371/WebSite2/Default.aspx>
- **Local IIS:** Carpeta de l'assetjo web per defecte (<http://localhost>)
  - Por ejemplo C:\Inetpub\wwwroot\WebSite1

# Code-inline vs Code-behind

- ▶ “Etiquetes” declaratives
  - HTML, controls de servidor, text estàtic
- ▶ A diferència d'ASP, bona separació entre el codi i les etiquetes

Únic arxiu  
 (“Code-inline”)

**código**

<etiquetas>

Form1.aspx

Arxius separats (“Code-behind”)

<etiquetas>

Form1.aspx

**código**

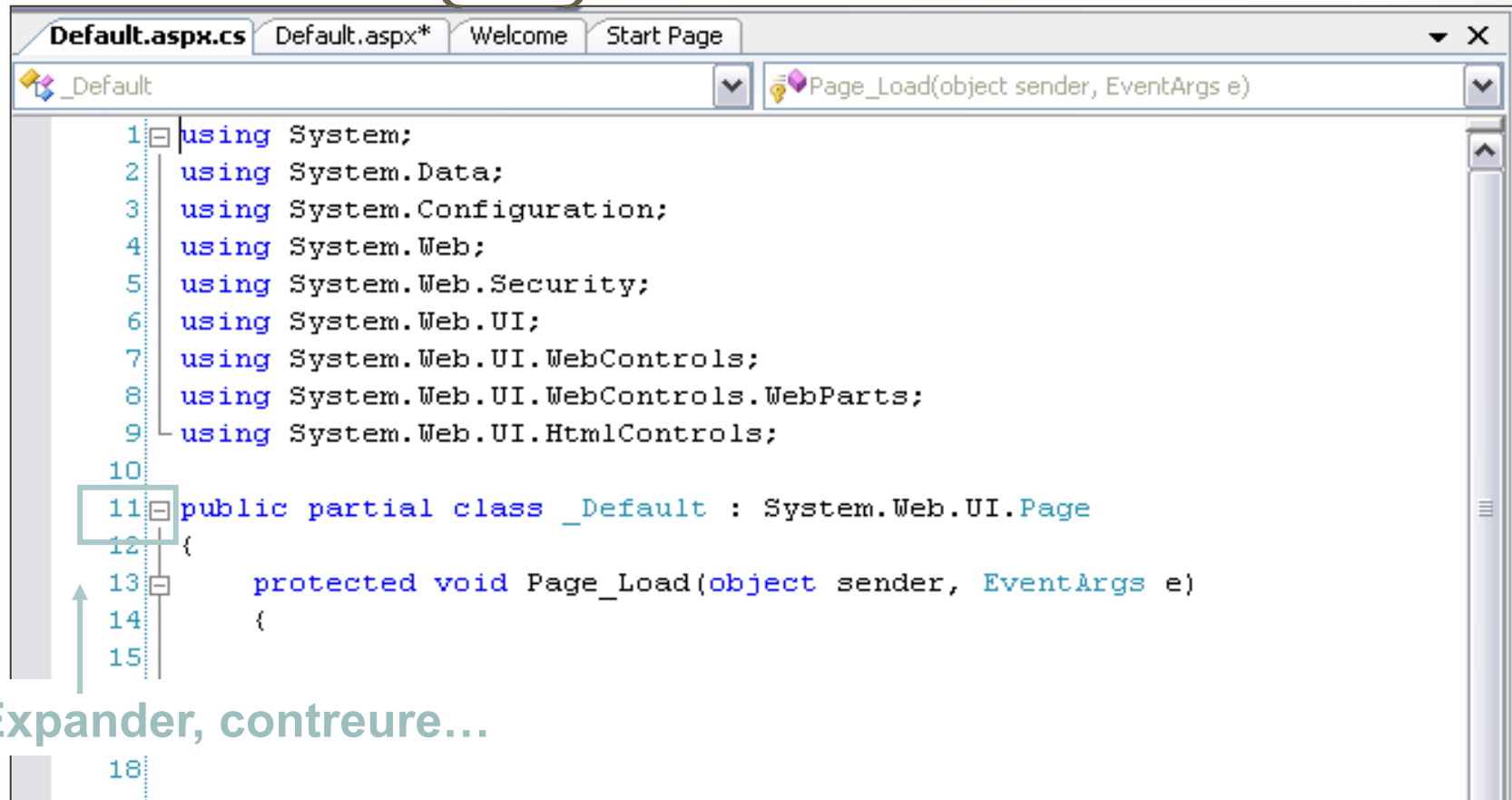
Form1.cs



# Code-behind

- El codi per manejar esdeveniments se situa en un **arxiu físic separat** de la pàgina que conté els controls de servidor i les etiquetes.
  - Extensió *.aspx*
  - Extensió *.cs*, *.vb* ... (*code-behind*)
- UTIL mantenir-ho per separat :  
És comú en grups de projectes tenir
  - dissenyadors treballant en l'IU de l'aplicació
  - i desenvolupadors en el comportament o codi.

# Veure codi (C#)



```
1 using System;
2 using System.Data;
3 using System.Configuration;
4 using System.Web;
5 using System.Web.Security;
6 using System.Web.UI;
7 using System.Web.UI.WebControls;
8 using System.Web.UI.WebControls.WebParts;
9 using System.Web.UI.HtmlControls;
10
11 public partial class _Default : System.Web.UI.Page
12 {
13     protected void Page_Load(object sender, EventArgs e)
14     {
15
16
17
18
```

Expander, contreure...

**Definició d'una classe parcial**

**Procediment que respon a l'esdeveniment LLOEU de la nàvina**

Cuadro de herra... Web.config

**Estándar**

- Puntero
- Label
- TextBox
- Button
- LinkButton
- ImageButton
- HyperLink
- DropDownList
- ListBox
- CheckBox
- CheckBoxList
- RadioButton
- RadioButtonList
- Image
- ImageMap
- Table
- BulletedList
- HiddenField
- Literal
- Calendar

Web.config Default.aspx\* Default.aspx\* Página de inicio Ex

**Propiedades**

**Button1** System.Web.UI.WebControls.Button

(Expressions)	
(ID)	<b>Button1</b>
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	<b>Button</b>
ToolTip	
UseSubmitBehavior	True
ValidationGroup	
Visible	True

**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

Lista de errores 0 errores 0 advertencias 0 mensajes

Descripción

Lista de errores Resultados

Listo

# Indice

1. Pàgines mestres
2. Controls de servidor
3. Esdeveniments
4. Maquetació amb CSS
5. Navegació entre WebForms

***1***

Pàgines mestres

# Pàgina mestra

- Les pàgines mestres permeten crear un disseny coherent per a les pàgines de l'aplicació.
- Es pot definir l'aspecte, el disseny i el comportament estàndard que desitja que tinguin totes les pàgines (o un grup de pàgines) de l'aplicació en una sola pàgina mestra.
- A continuació, es creen pàgines de contingut individuals que incloguin el contingut que desitja mostrar.

# En sol·licitar una pàgina...

- Quan els usuaris sol·liciten les pàgines de contingut, aquestes són combinades amb la pàgina mestra amb la finalitat de generar una sortida que combini el disseny de la pàgina mestra amb el de la pàgina de contingut.

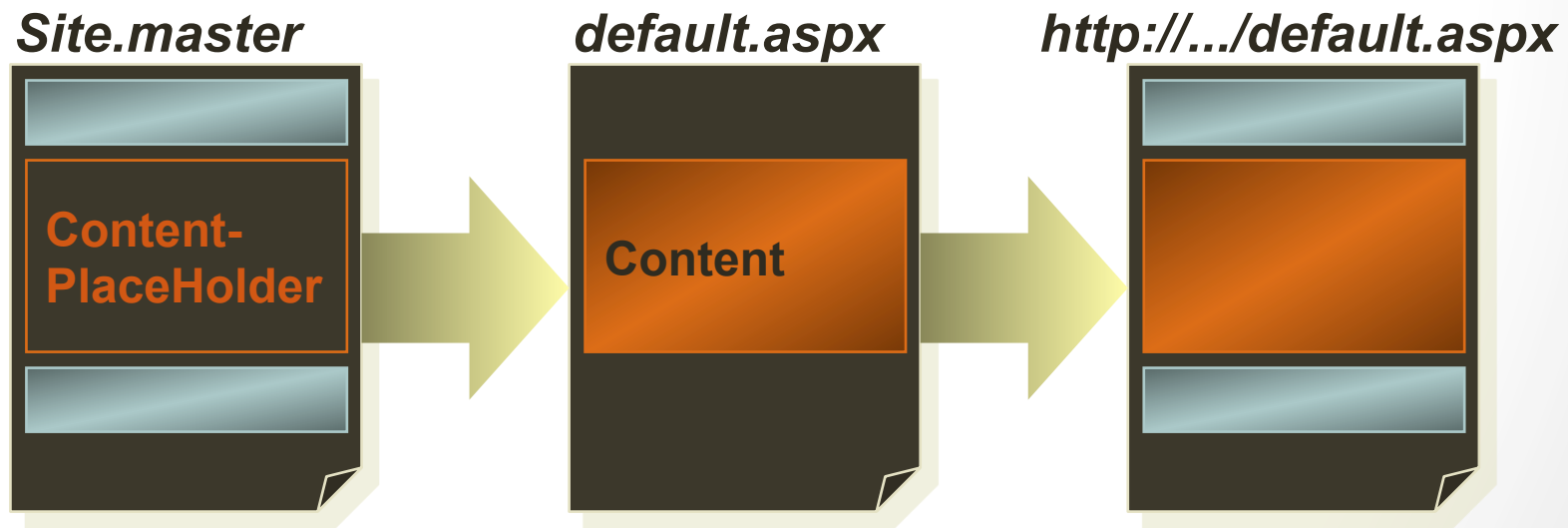
# Avantatges

- Realització de canvis de disseny en una sola ubicació; els canvis es veuran reflectits en totes les pàgines que usen la pàgina mestra.
- Reutilització de la interfície d'usuari
- Millor experiència de l'usuari final: pàgines més coherents



# Com funciona...

- Master pages defineixen el contingut comú i els contenidors de contingut (content placeholders)
- Content pages fan referència a les paginas mestres i omplen als contenidors amb el seu contingut.



# En temps d'execució

- IIS controla les pàgines mestres en la seqüència següent:
  - Els usuaris sol·liciten una pàgina escrivint l'adreça URL de la pàgina de contingut.
  - Quan es captura la pàgina, es llegeix la **directiva @ Page**. Si la directiva fa referència a una pàgina mestra, també es llegeix la pàgina mestra. Si les pàgines se sol·liciten per primera vegada, es compilen les dues pàgines.
  - La pàgina mestra amb el contingut actualitzat es combina en l'arbre de control de la pàgina de contingut.
  - El contingut dels controls Content individuals es combina en el control **contentplaceholder** corresponent de la pàgina mestra.
  - La pàgina combinada resultant es representa en l'explorador.

# Page.Master

- Nova propietat (Master) de `SYSTEM.WEB.UI.Page`
- Esta propietat conté una referència a la pàgina mestra de la pàgina de contingut, per tant proveeix a una pàgina contingut d'accés programàtic a la pàgina mestra
  - Determina si la pàgina té associada una mestra
  - Accés als controls definits en la mestra podent escriure codi suportat a les pàgines de contingut
  - Accés a mètodes públics i propietats definides en la mestra
- Integració a nivell codi de les pàgines mestres i continguts

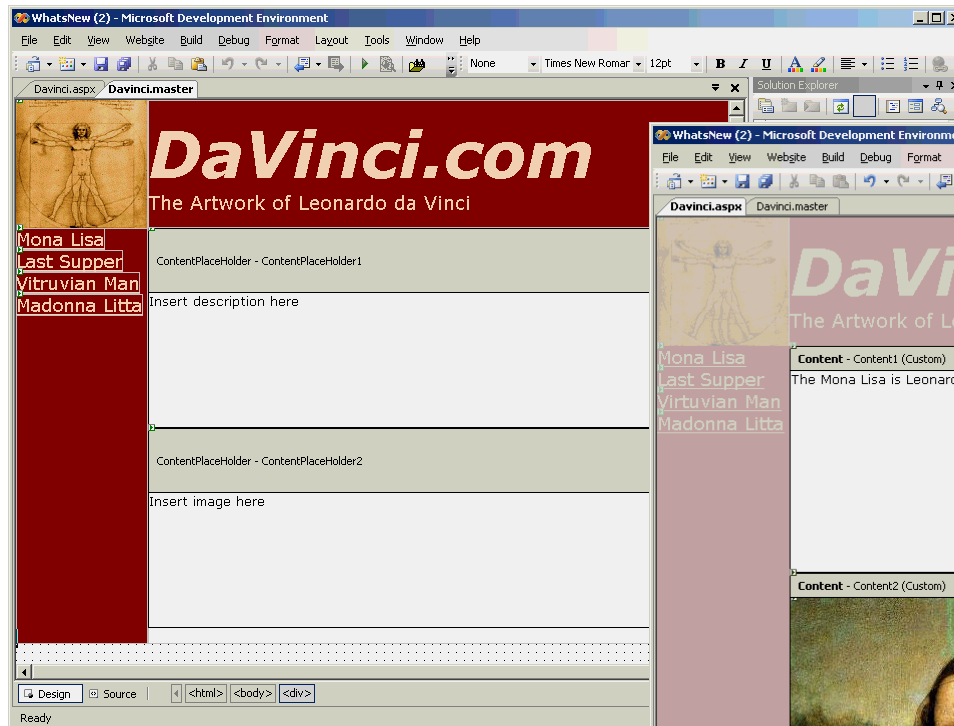
# Des del navegador..

- Des de la perspectiva de l'usuari, la combinació de les pàgines mestres i les pàgines de contingut dona com resultat una única pàgina. L'adreça URL d'aquesta pàgina és la de la pàgina de contingut.

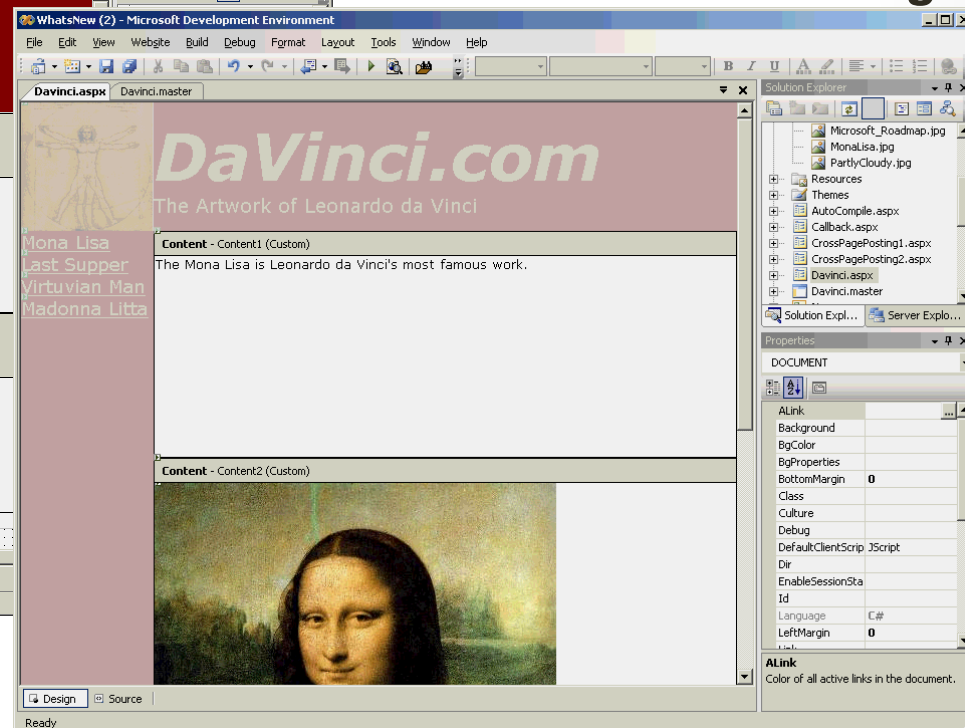
# En Visual Studio...

- Herència visual

## Master Page



## Content Page



2

Controls de  
servidor

# Els controls de servidor...

- Tenen propietats que poden ser establertes
  - declarativament (en l'etiqueta)
  - o mitjançant programació (en el codi).
- Juntament amb la pàgina, **tenen esdeveniments** que els desenvolupadors poden manejar
  - per executar accions específiques durant l'execució de la pàgina
  - o en resposta a una acció del costat del client que envia la pàgina al servidor.

# Controls de servidor

- Suport per a característiques avançades: enllaç de dades, plantilles..
- S'empra el prefix **asp:** juntament amb l'atribut **runat="server"**.  
`<asp:TextBox id="text" runat="server" />`



# Tipus de controls ASP.NET

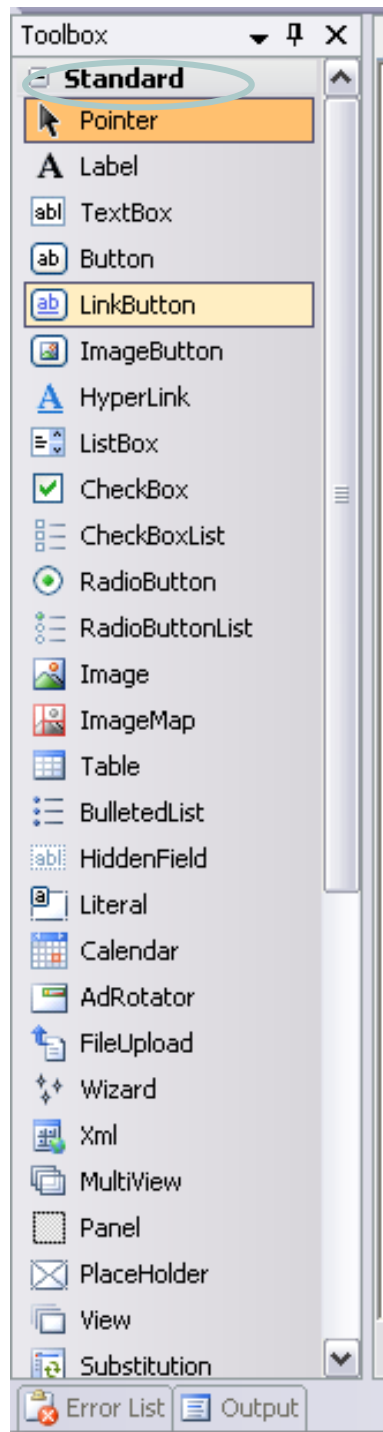
- 1. Controls HTML
- 2. Controls estàndard (web)
- 3. Controls de validació
- 4. Controls de login
- 5. Controls de navegació
- 6. Controls Webparts
- 7. Controls de dades
- 8. Controls d'usuari

## Principals controls web (estàndard)

Label	S'utilitza per mostrar text dinàmic (canviem les seves propietats a través del codi del servidor)
Hyperlink	Mostra un enllaç a una altra pàgina.
TextBox	Permet introduir text a l'usuari. Propietat Textmode valors: Single, Multiline o Password.
Image	Serveix per mostrar una imatge a la pàgina web.
Button	S'usa en un Formulari web per crear un control de tipus submit (esdeveniment OnClick) o un control de comando tipus botó (esdeveniment OnCommand).
LinkButton	Aparença de hipervínculo però funcions de control d'un botó (envio formulari)
ImageButton	Mostra una imatge que maneja esdeveniments tipus clic.
Checkbox	Serveixen per afegir caselles de verificació a una pàgina.
RadioButton	Crea un botó de ràdio individual a la pàgina.

# Principals controls web

DropDownList	Proporciona un bon mètode perquè els usuaris triïn elements d'una llista en un espai petit. Cada element es crea amb un control ListItem.
ListBox	Propietat SelectionMode: Single, Multiple. Cada element es crea amb un control ListItem.
CheckBoxList	Permet presentar una llista d'opcions podent l'usuari seleccionar varies.
RadioButtonList	Permet crear una llista de botons de radi d'opcions excloents.
Panel	Pot usar-se com a contenidor d'altres controls.
Table, TableRow, TableCell	Permeten crear dinàmicament una taula mitjançant programació.



## Controls estàndard

- 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>
```

Username:

Password:

Comments:

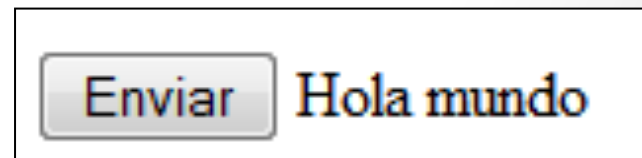
# Button

## Pàgina aspx

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

## Pàgina aspx.cs

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



# ImageButton

## Pàgina aspx

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

## Pàgina aspx.cs

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



# Panel

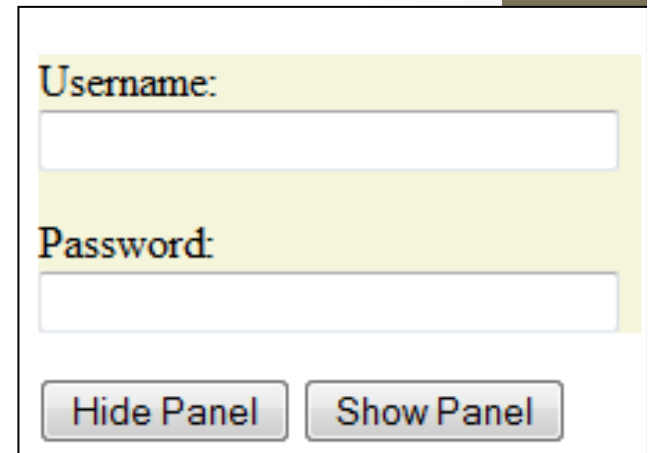
## Pàgina aspx

```
<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>
```

## Pàgina aspx.cs

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

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



The screenshot shows a web application interface. At the top, there is a light yellow rectangular panel. Inside this panel, the text "Username:" is followed by a text input field. Below that, the text "Password:" is followed by a password input field. At the bottom of the panel, there are two buttons: "Hide Panel" and "Show Panel". The "Show Panel" button is currently visible, indicating the panel is in its default state.



# 3

## Esdeveniments dels controls de servidor

# Model d'esdeveniments

- A les pàgines Web ASP.NET, els esdeveniments associats als controls de servidor s'originen en el client (explorador) però els controla la pàgina ASP.NET en el servidor Web.
- La informació de l'esdeveniment es captura en el client i es transmet un missatge d'esdeveniment al servidor mitjançant un enviament HTTP.
- La pàgina ha d'interpretar l'enviament per determinar l'esdeveniment ocorregut i, a continuació, cridar al mètode apropiat del codi del servidor per controlar aquest esdeveniment.

# Enllaçar esdeveniments a mètodes

- Un esdeveniment és un missatge que envia un objecte quan ocorre una acció (Ex. "s'ha fet clic en un botó") . L'acció pot estar causada per la interacció de l'usuari, com un clic, o per una altra lògica del programa.
- En una aplicació, s'ha de traduir el missatge en una trucada a un mètode del codi.
- L'enllaç entre el missatge de l'esdeveniment i un mètode específic (és a dir, un controlador d'esdeveniment) es duu a terme utilitzant **un delegat d'esdeveniments**.

# Esdeveniments i delegats

- L'objecte que provoca l'esdeveniment es coneix com a remitent de l'esdeveniment. L'objecte que captura l'esdeveniment i respon a ell es denomina receptor de l'esdeveniment.
- En les comunicacions d'esdeveniments, el remitent de l'esdeveniment no sap què objecte o mètode rebrà els esdeveniments que provoca. **Es necessita un intermediari (o mecanisme de tipus capdavanter) entre l'origen i el receptor.**
- .NET Framework defineix un tipus especial (**Delegate**) que proporciona la funcionalitat d'un punter a funció.

# Esdeveniments i delegats

- Un delegat és una classe que pot guardar una referència a un mètode.
- A diferència d'altres classes, una classe de delegat té un prototip i pot guardar referències únicament als mètodes que coincideixen amb el seu prototip.
- Per tant, un delegat equival a un punter a funció amb seguretat.
- Per convenció, els delegats d'esdeveniment de .NET Framework tenen dos paràmetres, l'origen que va provocar l'esdeveniment i les dades de l'esdeveniment

# Esdeveniments i delegats

- Els delegats d'esdeveniment personalitzats només són necessaris quan un esdeveniment genera dades d'esdeveniment. Molts esdeveniments, inclosos alguns esdeveniments d'interfície d'usuari, com els clics, no generen dades d'esdeveniment.
- En aquests casos, és apropiat el delegat proporcionat a la biblioteca de classes per a l'esdeveniment sense dades, [System.EventHandler](#).

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

- Els delegats d'esdeveniment són de multidifusió, la qual cosa significa que poden guardar referències a més d'un mètode de control d'esdeveniments. **A les pàgines Web ASP.NET, no és necessari codificar explícitament delegats si el control es crea mitjançant declaració (en el marcat) a la pàgina.**

# Manejadores d'esdeveniments

- El prototip dels mètodes manejadores d'esdeveniments han de coincidir amb el prototip del *delegat EventHandler*.
- Per tant, els manejadores d'esdeveniment en ASPnet retornen *void* i tenen **dos paràmetres**:
  - *Objecte que llança l'esdeveniment*
  - *Arguments de l'esdeveniment: informació específica de l'esdeveniment (EventArgs o tipus derivat)*
- Per exemple, per a un control ImageButton de servidor Web, el segon argument és de tipus ImageClickEventArgs, que inclou informació sobre les coordenades on l'usuari ha fet clic..

# Associar el manejador al control

- **Escriure-ho manualment**

## Arxiu Default.aspx

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

## Archivo Default.aspx.cs

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

- **Prémer doble clic sobre un control en vista dissenyo (esdeveniment per defecte)**

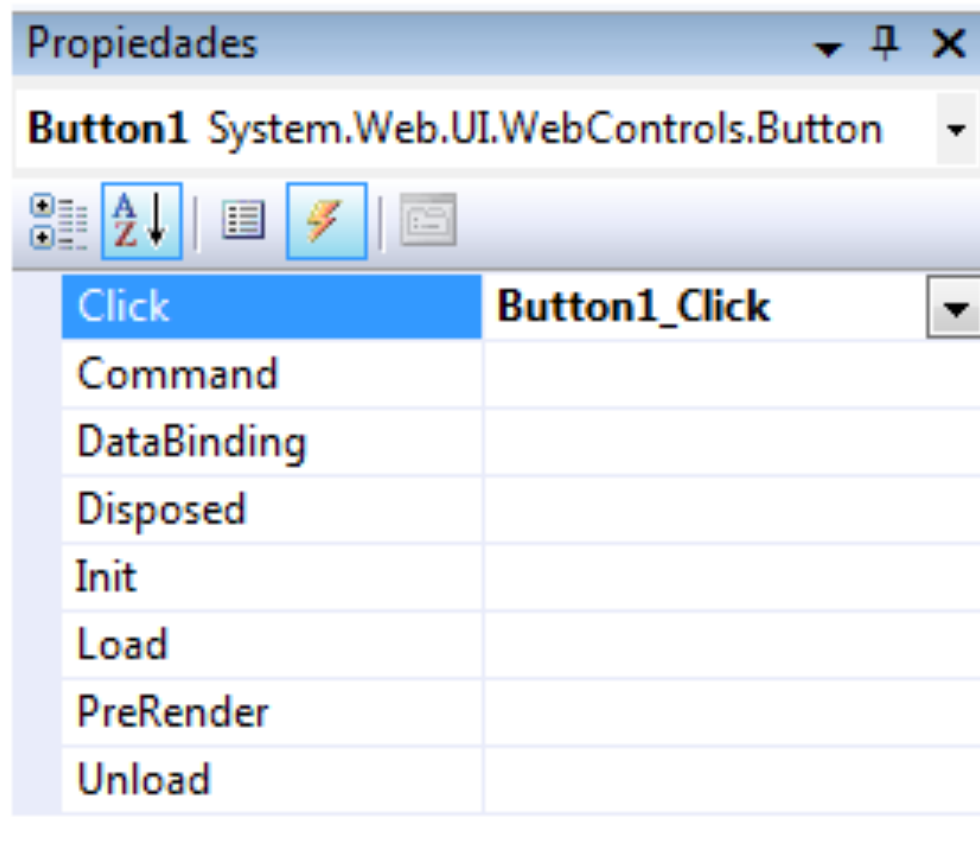
```
protected void Button1_Click(object sender,
  EventArgs e)
```

En compilar la pàgina, ASP.NET busca un mètode denominat EventoClick i confirma que aquest té la signatura adequada (accepta dos arguments, un de tipus **Object** i un altre de tipus EventArgs). A continuació, ASP.NET enllaça automàticament l'esdeveniment al mètode



# Associar el manejador al control

- Seleccionar l'esdeveniment de la finestra de propietats del control (i associar un nom de mètode manejador de l'esdeveniment o doble clic)



# Per crear un controlador d'esdeveniments en temps d'execució amb Visual C#

- Normalment això es fa quan s'estan creant controls mitjançant programació.
- Para això **s'ha de crear una instància del delegat EventHandler**, a la qual s'ha de passar l'adreça del mètode al que es va a enllaçar.
- Dispués és necessari agregar l'objecte delegat a la llista de mètodes als quals es diu quan es produeix l'esdeveniment.
- En l'exemple de codi següent es mostra la manera d'enllaçar l'esdeveniment **Clic** del control Button1 a un mètode denominat myEventHandler:

```
Button1.Clic += new System.EventHandler  
(this.myEventHandler);
```

# Example

```
Button b = new Button;  
b.Text = "Click";  
b.Click += new System.EventHandler(ButtonClick);  
Placeholder1.Controls.Add(b);
```

# Tipus d'esdeveniments

- **Esdeveniments d'enviament (POSTBACK).** Són els esdeveniments llançats per controls que emeten un enviament (post) al servidor de manera immediata per processar el Web Form.
  - Per exemple en enviar un formulari prement sobre el botó d'acceptació. Aquests controls són: Button, Link Button i Image Button.
- **Esdeveniments de caché (NO-POSTBACK).** Aquests esdeveniments es produeixen en la vista i seran processats en el servidor quan s'envii la informació mitjançant un esdeveniment d'enviament.
  - Per exemple el seleccionar un element d'una llista provoca un canvi d'estat de la mateixa que després, en servidor, podrem obtenir. Controls amb esdeveniments de caché són TextBox, DropDownList, CheckBox...

# Esdeveniments postback vs no-postback

- Una pàgina es carrega després de cada petició: fenomen conegut com **PostBack (=enviament)**.
- Esdeveniments **PostBack**:
  - causen que la informació del formulari s'envii al servidor immediatament.
- Esdeveniments **no-PostBack (o cached)**:
  - la informació s'envia en el següent esdeveniment **postback**.
  - Els esdeveniments es guarden en una cua en el client fins que un esdeveniment **postback** ocorre.

## Esdeveniments postback vs no-postback (II)

- Button, Link Button i Image Button causen esdeveniments **postback**.
- TextBox, DropDownList, ListBox, RadioButton i CheckBox, proveeixen esdeveniments **cached**.
  - No obstant això podem sobrecarregar aquest comportament en els controls per poder realitzar esdeveniments postback, canviant la propietat **AutoPostBack a true**.

# Connectar diversos esdeveniments amb un únic controlador d'esdeveniments

```
<asp:Button ID="Button1" onclick="Button_Click" runat="server"
  Text="Button1" />
<br />
<asp:Button ID="Button2" onclick="Button_Click" runat="server"
  Text="Button2" />
```

- **Per determinar el control que va provocar l'esdeveniment**

```
private void Button_Click(object sender, System.EventArgs e)
{
    Button b = (Button) sender;
    Label1.Text = b.ID;
}
```

es mostra el controlador de l'esdeveniment **Clic** d'un control **Button** al que criden diversos botons diferents. El controlador mostra la propietat **ANEU** del botó que va provocar l'esdeveniment.

# Esdeveniments de pàgina

- Les pàgines ASP.NET provoquen esdeveniments de cicles de vida com **Init**, **Lloeu**, **PreRender** i uns altres.
- De manera predeterminada, els esdeveniments de pàgina es poden enllaçar als mètodes utilitzant la convenció de nomenclatura **Page\_nombreDeEvento**.
  - Per exemple, amb la finalitat de crear un controlador per a l'esdeveniment **Lloeu** de la pàgina, es pot crear un mètode denominat **Page\_Lloeu**.
  - En temps de compilació, ASP.NET buscarà els mètodes que es basin en aquesta convenció de nomenclatura i realitzarà l'enllaç automàticament entre l'esdeveniment i el mètode.
- Es pot utilitzar la convenció **Page\_nombreDeEvento** per a qualsevol esdeveniment exposat per la classe **Page**.



# Esdeveniments de pàgina

- Les pàgines ASP.NET enllacen automàticament els esdeveniments de pàgines als mètodes que tenen el nom **Page\_esdeveniment**.
- Aquest enllaç automàtic ho configura l'atribut **AutoEventWireup** de la directiva @ Page, el valor de la qual s'estableix de forma predeterminada en **true**.
- Si estableix **AutoEventWireup** en **false**, la pàgina no busca automàticament els mètodes que usin la convenció de nomenclatura **Page\_esdeveniment**. Per tant, es poden crear els mètodes amb qualsevol nom i enllaçar-los als esdeveniments de pàgines explícitament.

# Propietat IsPostBack

- Una pàgina es carrega després de cada petició: fenomen conegut com **PostBack (=enviament)**.
- L'esdeveniment Page\_Load es produeix en cada petició.
- **Page.IsPostBack** per executar codi condicional

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

**Important:** Els mètodes de control d'esdeveniments de pàgina no requereixen cap argument. Aquests esdeveniments (com Load) poden acceptar els dos arguments estàndard, però en aquests arguments no es passa cap valor.

# 4

## Maquetació amb CSS

# Què és CSS?

- CSS és una eina que permet definir la presentació d'un document
- Permet crear un conjunt d'estils en una única localització
- Les pàgines a les quals s'aplica la mateixa fulla d'estils tindran les mateixes fonts, colors i grandària
- Proporcionen una estètica homogènia en totes les pàgines d'un lloc web

# Tipus de fulles d'estils

- Fulla d'estil externa
  - S'emplacen totes les regles d'estil en una única fulla d'estils externa
  - S'enllaça aquesta fulla d'estils externa amb totes les pàgines que vagin a tenir la mateixa aparença
  - En un formulari web:
    - `<link rel="Stylesheet" type="text/css" href="~/hoja.css" />`

Se situa dins de l'element HEAD

# Tipus de fulles d'estils

- Fulla d'estil interna
  - Fulla d'estil incrustada dins d'un document
  - Es defineixen les regles d'estil entre les etiquetes:

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

- Problema: S'ha de reescriure a cada pàgina

Se situa dins de l'element HEAD

# Tipus de fulles d'estils

- Fulla d'estil en línia
  - Permeten assignar un estil a un element determinat, usant el atribulo **style**

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

# Selectores d'estils

- En cas d'utilitzar fulles externes o internes cada regla d'estil té un selector
- Un selector és el tipus d'element al que es va a aplicar l'estil

- Exemple:

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

- En ASP.NET hi ha dos tipus de selectores:
  - Tipus d'elements i classes



# Selector de tipus element

- L'estil s'aplica a tots els elements del mateix tipus

```
h2 { color: #369;}
```

- Canvia de color totes les capçaleres de segon nivell

# Selector de tipus classe

- S'utilitza quan assignem a cada element una classe determinada

```
<p class="pageinfo">
```

```
    Copyright 2010
```

```
</p>
```

- La fulla d'estils contindrà per a cada classe una sèrie de característiques

```
.pageinfo
```

```
{
```

```
    font-family: Arial;
```

```
    font-size: x-small;
```

```
}
```

Els selectores de classe van precedits per un .

# Propietats d'estil

- **Font:** Tipus de font, grandària, color...
- **Background:** color de fons, imatge de fons...
- **Block:** espai entre paràgrafs, línies, paraules...
- **Box:** personalitzar taules, colors, vores...
- **Border:** dibuixa vores al voltant de diferents elements
- ...

# CssClass

- Asociar selector de tipus classe en Formularis Web en ASP.NET:

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

```
<title>Probando CSS</title>
```

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

```
</head>
```

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

hoja.css

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

# Maquetació amb CSS

- ***maquetar una pagina web és passar el disseny a codi HTML***, posant cada cosa en el seu lloc (una capçalera, un menu, etc.).
- Fins a fa uns anys l'única manera de maquetar una pagina web era mitjançant **taules HTML** (<table>), però això té molts desavantatges i limitacions
  - l'ús de les taules està condicionat a la mera tabulació de dades,
  - Un disseny amb taules no és flexible, és a dir, que no podem canviar la distribució dels elements a la pàgina, tret que la tornem a fer.
  - Cada Explorador renderiza de manera diferent cada document HTML i amb estructures amb taules el canvi és més notori
  - Ocupa més espai i més ample de banda.
  - Google no indexa d'igual manera les pàgines amb estructures basades en taules.
- Por això la tècnica de maquetació va ser evolucionant amb els anys fins a arribar al punt on no s'usen taules, si no capes (**els famosos DIVs**) a les quals se li donen format mitjançant CSS.

# Divs

- Les capes, layouts o divs són la mateixa cosa amb diferent nom, per tenir un concepte mental del que són, podem imaginar-los com a ***contenidors o blocs on podem ficar el que vulguem dins*** (imatges, text, animacions, un altre bloc, o tot al mateix temps) als quals se li assigna un ample, alt i posició, d'aquesta manera es van a anar posicionant aconseguint l'estructura que volem.

# Format a un DIV

- Per donar-li format a un DIV hem d'identificar-ho d'alguna forma, per a això existeix l'**atribut ANEU**, en el posarem el nom del DIV per després cridar-ho des de la fulla d'estils, la forma d'escriure-ho és així:

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

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

¡Esta es mi primer capa!

# Class o id?

- La diferència entre una classe (.) i un bloc (#) és que el bloc és únic i irrepetible en la pagina, és a dir, si vam crear un estil de bloc "**#busqueda**" per mostrar el quadre de cerca no podrem usar-ho una altra vegada en la mateixa pagina, en canvi si a "**#busqueda**" ho convertim en una classe "**.busqueda**" podrem usar-ho quantes vegades vulguem.



# Propietats de maquetació

```
<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>
```

- Els elements DIV poden centrar-se utilitzant els atributs **margin-left: auto;** **margin-right: auto;**
- La propietat **float** ajusta els elements cap al marge indicat (quan tenim capes adjacents).
- De vegades necessitaràs tenir una capa o bloc que no tingui capes a la seva/s costats, per a això aquesta la propietat CSS **Clear**. (ej el peu de pàgina). Aquesta propietat s'utilitza en conjunt amb *float* i **serveix per evitar que una capa es posicioni a qualsevol dels costats**

# Cuidat

- En les estructures clàssiques amb taules, podíem utilitzar grandàries en percentatge. Encara que aquí, també podem utilitzar percentatges, el dibuixat de la pàgina pot no veure's com s'esperava.

# Maquetar amb CSS links

- <http://www.comocrearunsitioweb.com/conceptos-basicos-css>
- <http://www.webexperto.com/articulos/art/232/por-que-maquetar-con-estandares/>
- <http://www.desarrolloweb.com/manuales/manual-css-hojas-de-estilo.html>
- <http://www.comocrearunsitioweb.com/maquetando-pagina-web-css>

5

# Navegació entre formularis

# Navegació tradicional

- En HTML:
  - Element `<a>` i atribut `href`
  - `<a runat="server" href="Login.aspx">Regístrate aquí</a>`
  - La pàgina `Login.aspx` ha d'estar en el mateix directori que la pàgina que conté l'enllaç
  - Després de prémer sobre l'enllaç es mostra la pàgina `Login.aspx`
- En ASPx:
  - `<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Login.aspx">Regístrate aquí</asp:HyperLink>`

# Diferents formes de navegar

- **Control hipervínculo**
  - Navega a una altra pàgina
- **Mètode `Response.Redirect`**
  - Navega a una altra pàgina per mitjà del codi. (Equivalent a navegar a través d'un enllaç)

# Hipervínculos i Redirecció

- Els hipervínculos responen a esdeveniments clic mostrant la pàgina especificada en la propietat `NavigateURL` del control.
- Si vols capturar un clic en el codi, has d'usar els esdeveniments d'un `LinkButton` o `ImageButton` i utilitzar `Response.Redirect("SiguientePagina.aspx");`
- També podem passar paràmetres a la nova pàgina.

# Pas de paràmetres a una altra pàgina

- Pas de paràmetres en la URL.
  - Emmagatzema les dades en la col·lecció QueryString
  - Múltiples paràmetres separats per &

```
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

- Limitacions amb QueryString
  - Els caràcters utilitzats han de ser caràcters permesos en una URL
  - La informació és visible als usuaris en la barra del navegador
  - Els usuaris poden modificar la informació provocant errors inesperats
  - Molts navegadors imposen un límit en la longitud de la URL

# Paso de parámetros a otra página

- Caràcters especials:
  - & (per separar múltiple query strings)
  - + (alternativa per representar un espai)
  - # (especifica un marcador en una pàgina web)
- Solució:
  - Utilitzar els mètodes de la classe `HttpServerUtility` per codificar les dades

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

# Paso de parámetros a otra página

- El método `UrlEncode` reemplaza los caracteres especiales por secuencias de fuga

`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)

# Objecte Request

- **Request:** proporciona informació sobre la petició HTTP del client que ha provocat la càrrega de la pàgina actual.
  - Informació sobre el client (`Request.Browser`, `Request.Browser.IsMobileDevice`, `Request.browser.id`)
  - Cookies (`Request.Browser.Cookies`)
  - Paràmetres passats a la pàgina:  

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