

Estructura de un sitio ASP.NET

Estructura de un WebForm

Si observamos el código HTML de una página ASPX vemos que además de la sección **head y body**, el body siempre tiene un elemento **form**: las páginas Web ASP.NET siempre tiene un formulario (y sólo uno).

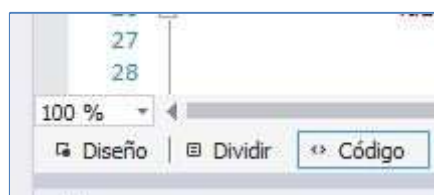
```
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7   <title></title>
8   <asp:ContentPlaceHolder ID="head" runat="server">
9     </asp:ContentPlaceHolder>
10 </head>
11 <body>
12   <form id="form1" runat="server">
13     <!-- Begin Wrapper -->
14     <div id="wrapper">
15
16       <!-- Begin Header -->
17       <div id="header" class="Encabezado">
18         ACME Computación & Electrónica
19       </div>
```

Por otro lado vemos que cada página ASP consta de 3 archivos, Por ejemplo si la página se llama pagina.aspx encontraremos:

- prueba.aspx
- prueba.aspx.cs
- prueba.aspx.designer.cs

pagina.aspx

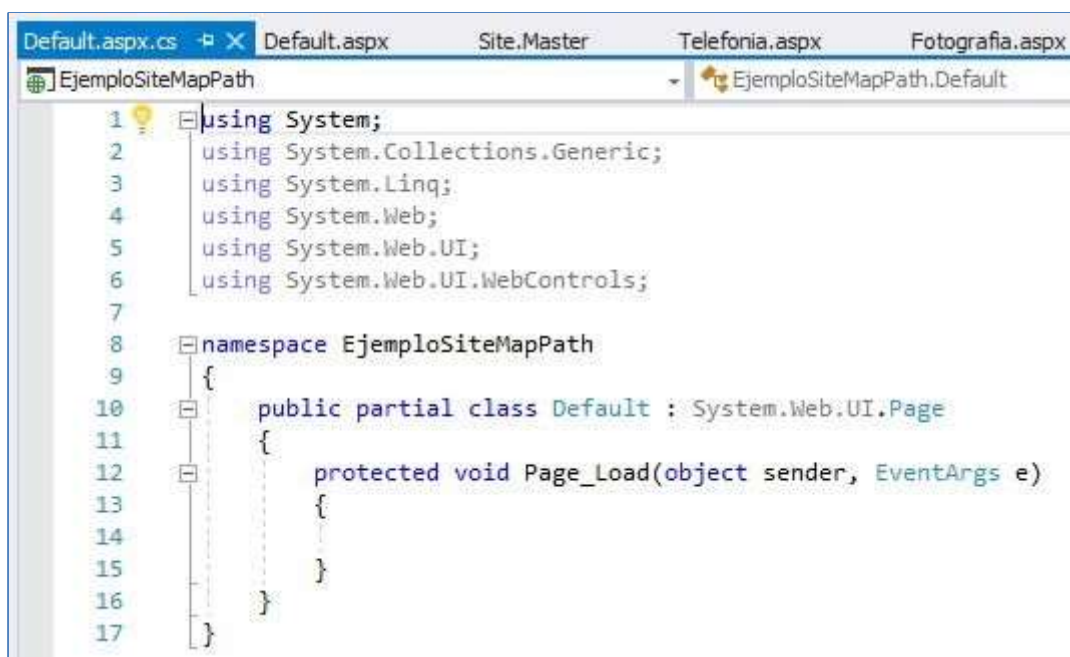
El archivo prueba.aspx contiene el código HTML de la página. Esta vista en Visual Studio se llama "**markup**" y a su vez puede verse de 3 formas diferentes: "**design**", "**split**", "**source**" (o sus nombres equivalentes en español según el lenguaje del IDE).



- La vista **design** muestra aproximadamente cómo se verá la página en el navegador Web.
- La vista **source** muestra sólo el código HTML.
- La vista **split** una combinación entre ambas.

prueba.aspx.cs

Este archivo contiene exclusivamente código C# (o VB.NET si elegimos crear la página en dicho lenguaje. En ese caso la extensión **cs** se reemplaza por **vb**).



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7
8 namespace EjemploSiteMapPath
9 {
10     public partial class Default : System.Web.UI.Page
11     {
12         protected void Page_Load(object sender, EventArgs e)
13         {
14         }
15     }
16 }
17 }
```

prueba.aspx.designer.cs

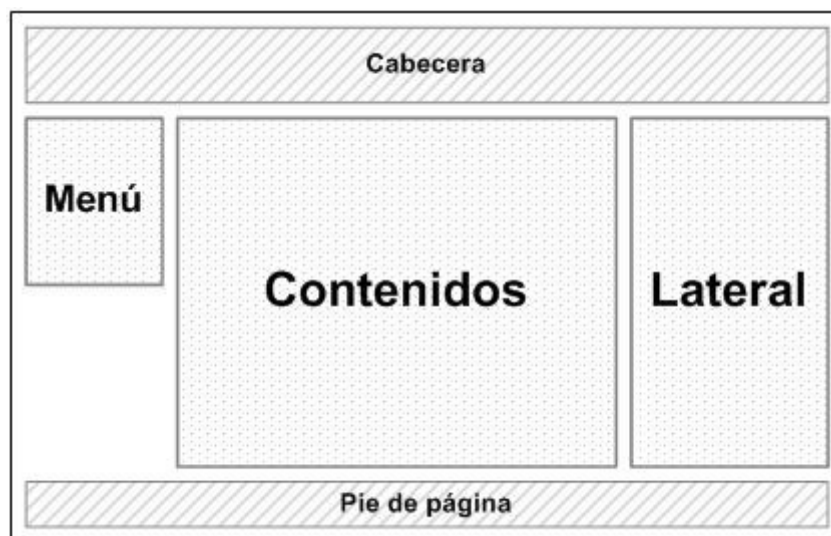
Este archivo contiene el código generado automáticamente por Visual Studio. **Normalmente no existen motivos para modificar dicho código.** Utiliza como en todo código NET una clase Partial que ayuda a separar código pero que compila como un único archivo al compilar.

```
Default.aspx.designer.cs  Default.aspx.cs  Default.aspx  Site.Master  Telefonía.aspx  Fotografía.aspx
EjemploSiteMapPath  EjemploSiteMapPath.Default

1  //-----
2  // <auto-generated>
3  //   This code was generated by a tool.
4  //
5  //   Changes to this file may cause incorrect behavior and will be lost if
6  //   the code is regenerated.
7  // </auto-generated>
8  //-----
9
10 namespace EjemploSiteMapPath
11 {
12
13
14     public partial class Default
15     {
16
17     }
18 }
```

Master Page

Los sitios Web suelen tener una estructura común en todas o al menos un grupo de sus páginas. Por ejemplo, vemos una ejemplo de esta estructura comun:

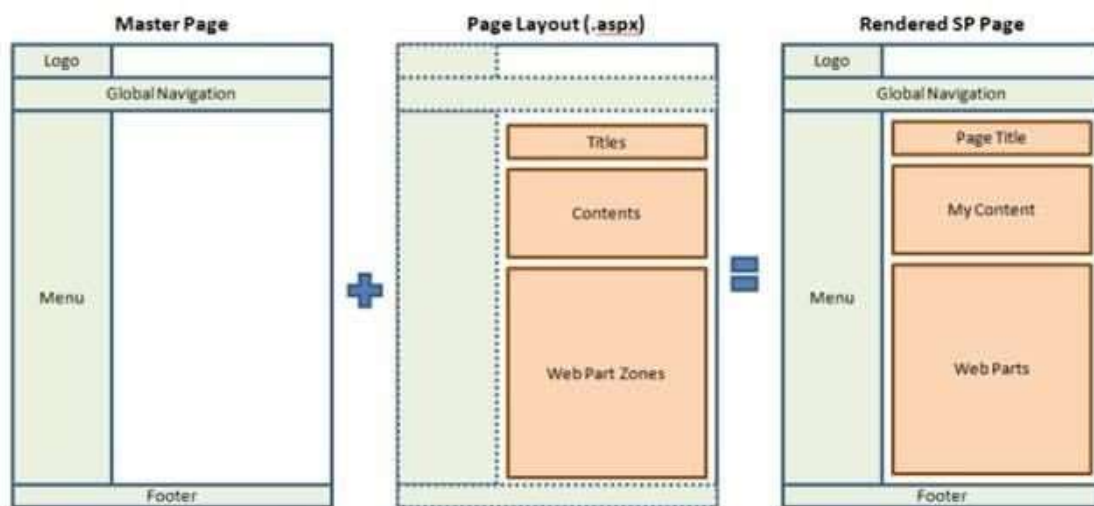


Si quisiéramos mantener por ejemplo este esquema en varias páginas deberíamos reproducir en cada una de ellas el encabezado, el menú, el pie de página, etc. Obviamente, si modificásemos uno de estos elementos deberíamos reproducir la modificación en todas las páginas del sitio. Esto no es óptimo ni cómodamente mantenible!

Para evitar este problema existe la denominada Master Page. El objetivo de la esta página maestra es proporcionar una estructura en común a todo el sitio. Cómo pueden coexistir varias Master Pages en un sitio, no es necesario que todas las páginas compartan el mismo esquema.

Si observamos el markup de la Master Page observamos que dentro de las secciones head y body existe un control de servidor llamado ContentPlaceholder. El propósito de este control es indicar dónde el motor de ASP.NET inyectará el contenido variable de cada página, y en una master page puede haber varios controles

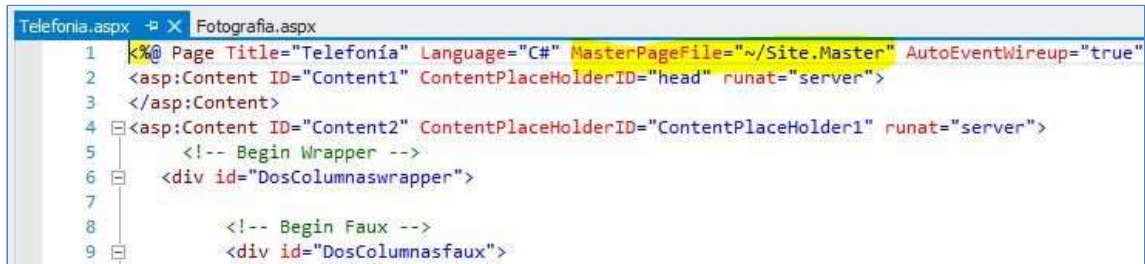
ContentPlaceholder. El resto de la Master Page es compartido por dichas páginas.



WebForm asociado a la Master Page

Hemos visto que la Master Page contiene los elementos head y body. Los Web Forms que hemos generado hasta el momento (ver. ej. HolaMundo) también contienen dichas secciones. Es por ello que existe un nuevo tipo de Web Form asociado a la Master Page (Web Form with Master Page).

Si observamos el markup veremos que ya no existen las secciones head y body y que además aparecen 2 controles de servidor del tipo Content. Todo lo que esté encerrado dentro de los elementos Content será inyectado en los respectivos Content Place Holder de la Master Page.



```
1 <%@ Page Title="Telefonia" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" %>
2 <asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
3 </asp:Content>
4 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
5 <!-- Begin Wrapper -->
6 <div id="DosColumnaswrapper">
7
8 <!-- Begin Faux -->
9 <div id="DosColumnasfaux">
```

Navegación

La navegación entre páginas en una aplicación ASP.NET puede establecerse de diferentes maneras.

Usando el elemento html anchor

Por ejemplo:

```
<a href="default.aspx">Inicio</a>
```

Por supuesto podemos hacer referencia a una página externa:

```
<a href="http://educacionit.com">EducacionIT</a>
```

Hay que tener presente que si utilizamos el elemento anchor las ubicaciones referenciadas a una página dentro de nuestro sitio son relativas. Por ejemplo, default.aspx será encontrada siempre que sea referenciada desde una página que se encuentre en la misma carpeta que default.aspx. Si lo hacemos desde otra carpeta, intentará buscarla en la misma y no la encontrará.

En el segundo ejemplo no se produce este problema ya que apunta a una dirección absoluta fuera de nuestro sitio.

Usando controles de servidor

Por ejemplo el **HyperLink**:

```
<asp:HyperLink ID="HyperLinkInicio" runat="server"
NavigateUrl="~/Default.aspx">HyperLink</asp:HyperLink>
```

La ventaja de utilizar este control (u otros controles de servidor) para referenciar páginas radica en que las referencias siempre son absolutas y además es posible configurar tanto la referencia cómo otras propiedades del control desde el código .NET:

```
this.HyperLinkInicio.NavigateUrl = "~/Default.aspx";
```

También es posible navegar entre páginas utilizando los controles Button, LinkButton e ImageButton.

Usando controles de navegación asociados al SiteMapPath

ASP.NET provee un mecanismo de navegación basado en un control de servidor llamado SiteMapPath.

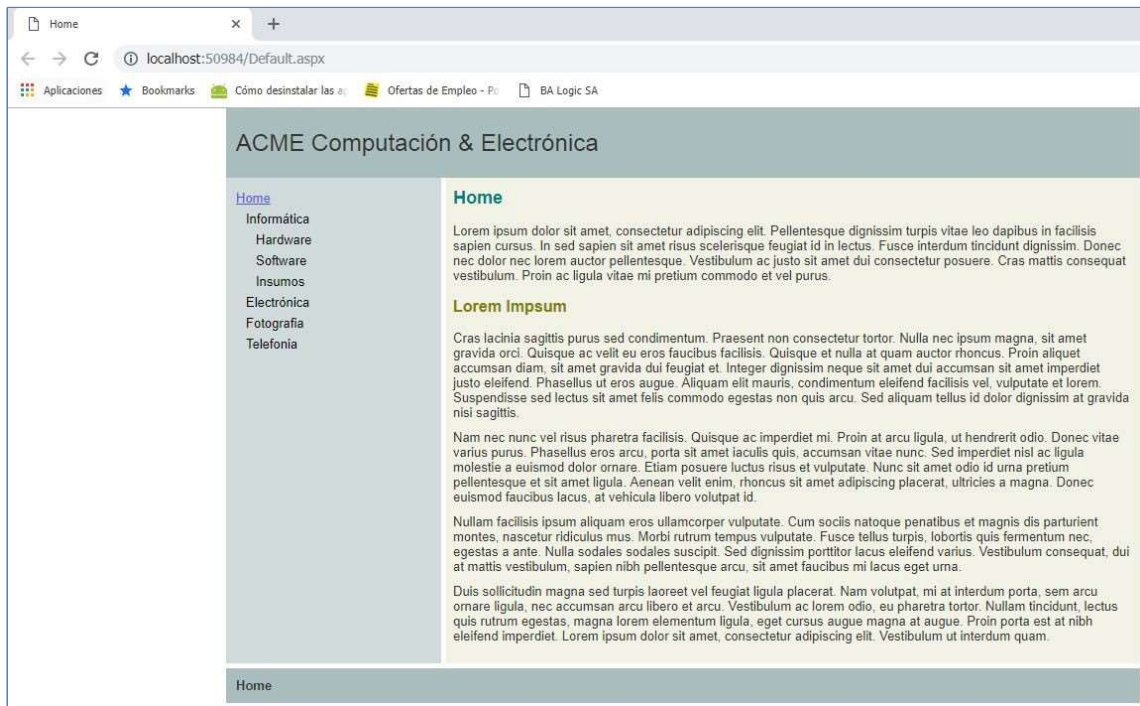
Este control es capaz de leer un archivo XML que contiene la estructura del sitio (Web.sitemap) y además provee una interfaz para que los controles Menu y TreeView representen automáticamente dicha estructura.

Por ejemplo:

```
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >    <
siteMapNode url="/~/default.aspx " title="Inicio" description="Va a la página
de inicio">
        <siteMapNode url="/~/ayuda.aspx " title="Ayuda" description="Va a la
página de ayuda" />
    </siteMapNode>
</siteMap>
```

Un aspecto a destacar es que los controles Menu y TreeView no son compatibles con Bootstrap, con lo cual muchos desarrolladores optan por utilizar los controles indicados en el punto anterior.

En la descarga EjemploSiteMapPath se puede ver cómo utilizar este control. Además incorpora el uso de CSS sin utilizar Bootstrap y el concepto de ASP.NET theme.



ASP.NET css + properties

Los controles de ASP.NET fueron diseñados para soportar diferentes formas de configurar su apariencia:

- propiedades del control
- CSS

Propiedades de control

Es la forma más fácil de cambiar la apariencia de un control. Por ejemplo:

```
<asp:Label ID="lblTitulo" runat="server" Text="Label"
ForeColor="Green"></asp:Label>
```

o también:

```
this.lblTitulo.ForeColor = System.Drawing.Color.Green;
```

Sin embargo la enorme desventaja de esto es que en caso de que modificar, por ejemplo, todos los títulos en nuestra aplicación deberemos modificar una a una todos los controles.

Css

La mayoría de los controles de servidor soportan a través de la propiedad `CssClass` la posibilidad de asociar una clase CSS a dicho control. La ventaja es que a través de la clase es posible definir varios elementos de la apariencia, no solamente uno como en el caso de las propiedades.

Por ejemplo:

```
<asp:Label ID="lblTitulo" runat="server" Text="Label"  
CssClass="alert">
```

Si luego deseamos cambiar la apariencia de varias propiedades al mismo tiempo basta con modificar la clase en cuestión.

La CSS puede asociarse a nivel de página o mejor aún, a nivel de Master Page:

```
<link href="Content/bootstrap.css" rel="stylesheet" />
```