Capítulo 9
Aplicando Master Pages, temas y estilos

Objetivo

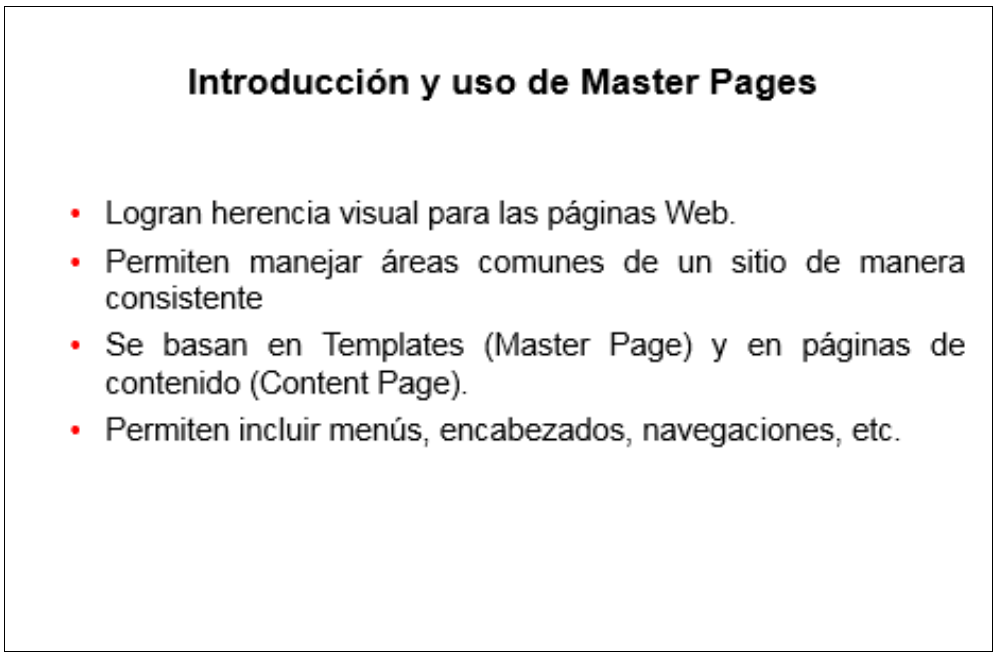
Al finalizar el capítulo, el alumno:

- Utilizar páginas maestras.
- Utilizar temas en aplicaciones Web.
- Aplicar estilos en páginas Web.

Temas

1. Introducción y uso de Master Pages
2. Uso de estilos
3. Aplicando estilos a los controles

1. Introducción y uso de Master Pages



El soporte de páginas maestras (MasterPages) fue una de las características más populares que se introdujeron con ASP.NET, y es una de esas características que casi todos los proyectos ASP.NET usan para mostrar una interfaz consistente en todo un sitio web.

Una página maestra es una página especial que proporciona marcas y controles para definir una estructura y unos elementos de interfaz comunes para el sitio, tales como la cabecera de página o la barra de navegación, en una ubicación común denominada "master page", para ser compartidos por varias páginas del sitio.

Uno de los escenarios más interesantes es el soporte que ASP.NET tiene para permitir jerarquizar páginas maestras, de tal forma que se puede crear una raíz master para un sitio web, y crear páginas maestras jerarquizadas basadas en la raíz y personalizarla (por ejemplo: se podría crear una master page **SingleColumn.Master** y otra **TwoColumn.Master** que definan dos estructuras generales de una y dos columnas para visualizar el sitio basadas en el template raíz). Esta jerarquía de páginas maestras es realmente flexible, y permite a los desarrolladores y diseñadores, realizar cambios rápidamente a la visualización y a la organización del sitio con cambios mínimos en el código y sin duplicación de contenidos.

Esto mejora la mantenibilidad del sitio y evita la duplicación innecesaria de código para estructuras o comportamientos del sitio que son compartidos. Por ejemplo, si todas las páginas deben tener los mismos banners de cabecera y pie de página o el mismo menú de navegación, se puede definir esto en una

Master Page una vez, de forma que todas las páginas asociadas a dicha Master Page heredarán estos elementos comunes. La ventaja de definir la cabecera, el pie de página y la navegación en una Master Page es que estos elementos solo tendrán que ser definidos una vez, en lugar de muchas veces y en código duplicado en las diferentes páginas del sitio.

A diferencia de los formularios Web, las páginas maestras no contienen la directiva @**Page**, en su lugar, contienen la directiva @**Master**.

Las páginas maestras permiten crear funcionalidad y diseño fáciles de mantener para múltiples páginas Web. Además, proporcionan funcionalidad que los desarrolladores normalmente creaban, a través de:

- Copiado y pegado de código, texto y elementos.
- Usando framesets.
- Utilizando controles ASP .NET.

Algunas de las ventajas de las páginas maestras son:

- Permiten centralizar la funcionalidad común de las páginas, de tal manera que el mantenimiento se realiza en un solo lugar.
- Permiten crear un conjunto de controles y código, y aplicarlos a diferentes páginas, de manera sencilla. Por ejemplo: se puede crear un menú basado en controles en una página maestra y aplicar el menú a todas las páginas deseadas.
- Proporcionan un modelo de objetos que permite acceder a elementos de la página maestra desde la página Web usada.

Master Pages y Content Pages

La definición de una Master Page es como la de cualquier página, pueden contener marcas, controles, código o cualquier combinación de estos elementos. Sin embargo, una Master Page puede contener un tipo especial de control llamado **ContentPlaceHolder**, el cual define una región de la representación de la master page que puede substituirse por el contenido de una página asociada a la maestra.

Además, un ContentPaceHolder también puede tener un contenido por defecto, por si la página derivada no necesita sobrescribir este contenido. La sintaxis de un control ContentPlaceHolder es como sigue.

```
<%-- Control ContentPlaceHolder--%>
<asp:contentplaceholder id="FlowerText" runat="server"/>

<%-- ContentPlaceHolder con contenido por defecto --%>
<asp:contentplaceholder id="Contentplaceholder1" runat="server">
  <h3>Welcome to my florist website!</h3>
</asp:contentplaceholder>
```

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

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

Una **Content Page** puede declarar controles **Content** que sobreescriben específicamente el contenido de las secciones marcadas en la Master Page. Un control Content, se asocia a un control ContentPlaceHolder particular a través de la propiedad **ContentPlaceHolderID**. Una Content Page, debe contener marcas y controles solo dentro de los controles Content; no puede tener ningún contenido de alto nivel por sí misma. Puede, sin embargo, tener directivas o código del lado del servidor.

Sección de Master Page con el placeholder que ocuparán las páginas:

```
<div class="container body-content">
    <asp:ContentPlaceHolder ID="MainContent" runat="server">
    </asp:ContentPlaceHolder>
    <hr />
    <footer>
        <p>&copy; <%: DateTime.Now.Year %> - My ASP.NET Application</p>
    </footer>
</div>
```

Sección de un Content Page que se relaciona con la Master Page a través del atributo ContentPlaceHolderID:

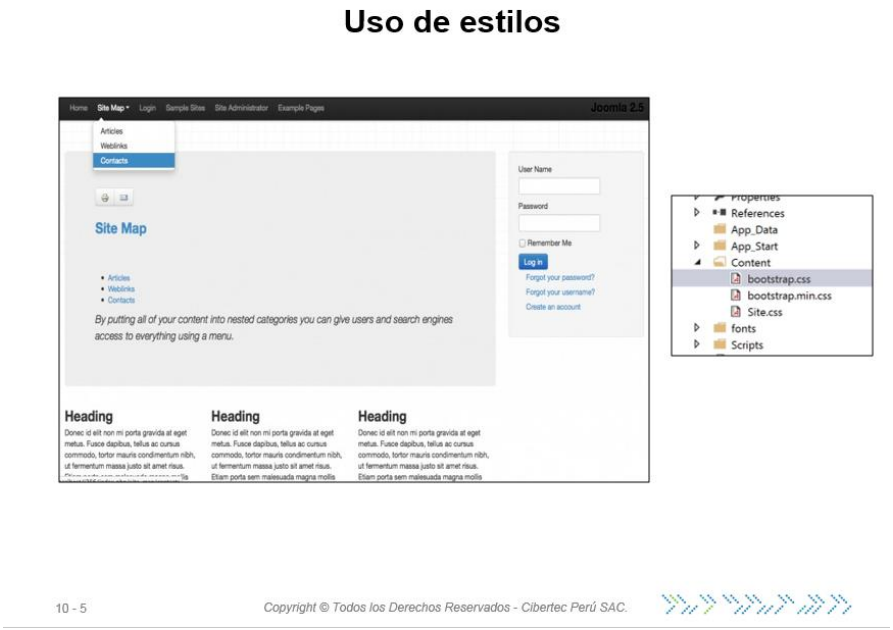
```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

    <div class="jumbotron">
        <h1>ASP.NET</h1>
        <p class="lead">ASP.NET is a free web framework for building great Web sites and
Web applications using HTML, CSS, and JavaScript.</p>
        <p><a href="http://www.asp.net" class="btn btn-primary btn-lg">Learn more
&raquo;</a></p>
    </div>

    <div class="row">
        <div class="col-md-4">
            <h2>Getting started</h2>
        </div>
    </div>

</asp:Content>
```

2. Uso de estilos



El modo predominante y estándar definido para modificar la presentación en una página Web es mediante Estilos. Estos estilos son definidos con archivo o archivos CSS que poseen una notación específica a aplicar sobre el HTML que definimos en nuestras páginas.

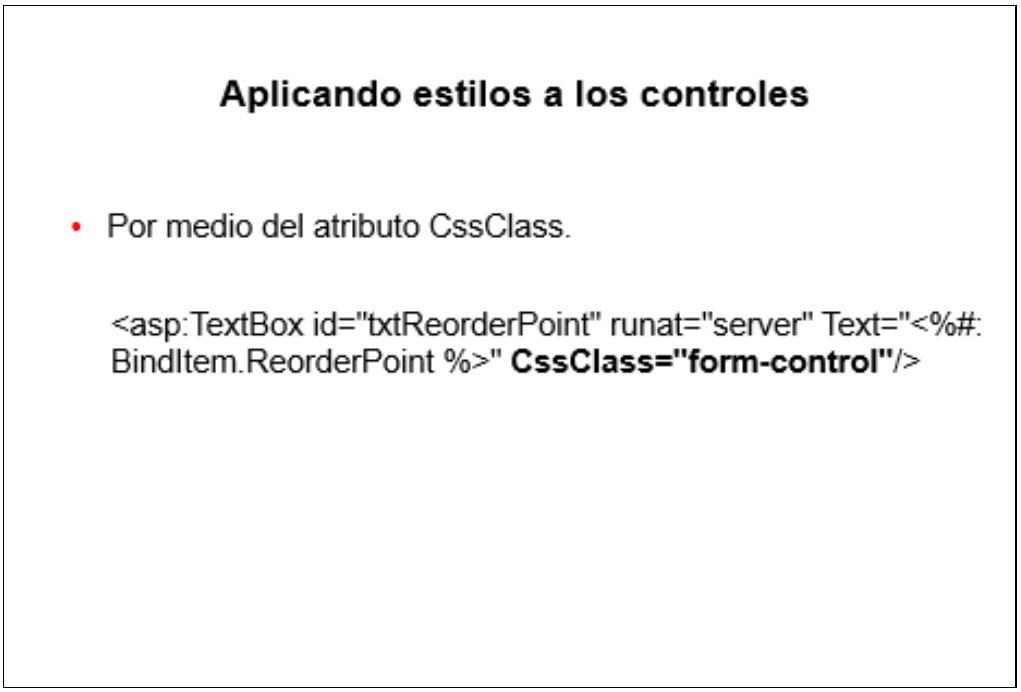
ASP.NET en general ha adoptado el uso de estilos llevando incluso una plantilla de estilos predeterminada que es bastante completa y popular, esta plantilla se llama Twitter Bootstrap. Para información detallada sobre la plantilla se puede consultar: http://getbootstrap.com/

En base a esta plantilla podemos hacer que los nuevos controles y elementos que se agreguen a una aplicación Web tomen un aspecto uniforme.

El estilo de estandarizado de Bootstrap y todos los estilos CSS por convención se ubican dentro de la carpeta Content, dentro de esta carpeta podemos crear subcarpetas según veamos conveniente para de esta manera organizar el contenido de múltiples archivos de estilos.

La plantilla por defecto de Twitter se puede personalizar para cambiar colores y fuentes. Para esto tenemos simplemente que reemplazar el archivo bootstrap.css por otro con la presentación que buscamos. Existen múltiples sitios y proyectos desde el cual podemos visualizar y descargar otros temas de Bootstrap. Por ejemplo, podemos consultar: http://bootswatch.com/

3. Aplicando estilos a los controles



Aplicar estilos a controles de WebForms es sencillo, solo basta con añadir un atributo CssClass al control que queremos personalizar y colocarle la clase que le correspondería al renderizarse en un elemento HTML. Por ejemplo, podemos asignarle el estilo **.form-control** a un control de tipo TextBox de la siguiente manera:

```
<asp:TextBox id="txtReorderPoint" runat="server" Text="<%#: BindItem.ReorderPoint %>" CssClass="form-control"/>
```

Lo que sucederá es que automáticamente se usarán la clase definida en el archivo bootstrap.css que tengamos para aplicar las modificaciones necesarias sobre el control.