

UNIVERSIDAD CATÓLICA DE SANTA MARÍA
PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

SESIÓN N° 03:

Controles Web

I

OBJETIVOS

- ❖ Conocer las características de los formularios web.
- ❖ Conocer la anatomía de una aplicación ASP.NET.
- ❖ Reconocer y utilizar las clases de controles HTML.
- ❖ Diseñar aplicaciones que corran del lado del servidor.

II

TEMAS A TRATAR

- ❖ Introducción.
- ❖ Controles Web
- ❖ CSS
- ❖ PSEUDOCLASE CSS
- ❖ Sass
- ❖ Resumen

III

MARCO TEORICO

1. INTRODUCCIÓN

En el mundo moderno el acceso a la información, así como los recursos de disposición y presentación de la información están fuertemente vinculados a dos medios esenciales, el primero es la internet y el segundo la computación móvil; integrados ya por el uso de los clientes, no hay fronteras claras para los usuarios ya que los dos se complementan y superponen, sin embargo, la programación sobre aplicaciones para internet supone el uso de recursos de programación que se integran a la presentación de aplicaciones sobre un navegador como cliente, los servidores web vienen a ser proveedores de recursos de presentación de información que tienen por tarea fundamental poder visualizar diferentes recursos de información multimedia y a la vez datos obtenidos de diferentes orígenes en tiempo real, de forma dinámica y con las posibilidades de interactuar con la aplicación.

2. CONTROLES WEB

- Los controles Web, con controles con muchas más características que los controles de servidor HTML:
 - *Proporcionan un interfaz de usuario enriquecida*
 - *Proporcionan un modelo de objetos consistentes*
 - *Detectan su salida automática*
 - *Proveen características de alto nivel*
- **Page.IsPostBack** (Propiedad): Obtiene un valor que indica si la página se está mostrando por primera vez o si se está cargando como respuesta a una devolución (postback). Es **true** si la página se carga como respuesta a un postback del cliente; en caso contrario, es **false**.

- control **CheckBoxList**: proporciona un grupo de casillas de selección múltiple que se pueden generar dinámicamente mediante el enlace de datos. Contiene una colección de **Items** con miembros que se corresponden a elementos individuales de la lista. Para determinar cuáles son los elementos seleccionados, recorra la colección en iteración y compruebe la propiedad **Selected** de cada elemento de la lista.
- control **BulletedList**: crea un control que genera una lista de elementos con formato de viñetas.
- control **HyperLink**: muestra un vínculo a otra página Web.
- control **MultiView**: es un contenedor para un grupo de controles **View**. Permite definir un grupo de controles View en el que cada control View contiene controles secundarios. A continuación, la aplicación puede representar un control View concreto en el cliente basándose en criterios de diferenciación.

3. CSS

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

4. PSEUDOCLEASE CSS

Una pseudoclase CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado. Por ejemplo, `:hover` aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

```
div:hover {  
    background-color: #F89B4D;  
}
```

Las pseudoclasses, junto con los pseudo-elementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como el historial del navegador (`:visited`, por ejemplo), el estado de su contenido (como `:checked` en algunos elementos de formulario), o la posición del ratón (como `:hover` que permite saber si el ratón está encima de un elemento o no).

Nota: En lugar de usar pseudoclasses, pseudo-elements puede usarse para dar estilo a una parte específica de un elemento.

Sintaxis

```
selector:pseudoclase { propiedad: valor; }
```

Al igual que las clases, se pueden concatenar la cantidad de pseudoclasses que se deseen en un selector.

5. SASS

Es un procesador CSS. Un preprocesador CSS es una herramienta que nos permite generar, de manera automática, hojas de estilo, añadiéndoles características que no tiene CSS, y que son propias de los lenguajes de programación, como pueden ser variables, funciones, selectores

anidados, herencia, etcétera.

A continuación, veremos un pequeño ejemplo de Sass, dónde podemos ver los elementos básicos de este preprocesador CSS, como estructuras repetitivas, variables, funciones o parámetros:

```
.row {
    @include flex_mixin:
}

/// Número de elementos máximos que voy a tener a lo ancho del layout
/// @group layout
$num_elementos: 8;

///Función que devuelve la anchura correspondiente al elemento
@function anchura_col($i) {
    @return (100 / $num_elementos)*$i ;
}

///Bucle para generar las clases para tamaños medios
@for $i from 1 through $num_elementos {
    .row > .col#{$i} {
        width: #{anchura_col($i)} + "%";
    }
}
```

Podemos ver que hay muchos elementos, pero tienen una sintaxis muy similar a CSS, añadiendo potencialidades de lenguaje de programación.

Una vez que ejecutamos nuestro fichero SCSS, se generaría un fichero CSS de salida de forma automática.

6. RESUMEN

La internet se inició en la tarea de presentar información de manera integrada con HTML, las páginas web eran estáticas inicialmente y mostraban la información como un diario o revista con mayor cantidad de medios de información integrados. La necesidad de interactuar con las páginas como si fueran aplicaciones llevo a evolucionar el lenguaje e incorporar Formularios HTML; el lenguaje HTML es el recurso de codificación de la información por excelencia en internet, a su vez la computación del lado del servidor se hace a través de diferentes lenguajes que realizan procesos complejos y el resultado de la computación la convierten en lenguaje HTML para luego enviarla al cliente que visualiza el resultado en un navegador web.

IV

(La práctica tiene una duración de 2 horas) ACTIVIDADES

1. EXPERIENCIA DE PRÁCTICA N° 01: FORMULARIO CON HTML

Integrarse al equipo conformado por el Jefe de Práctica.

Crear un espacio en GitHub para crear el proyecto involucrado en la siguiente práctica.

a) CONTROLES WEB

1. Crear una aplicación web y grabarla con el nombre de ControlWeb a su aplicación web
2. Agregue al formulario un control Label y configure sus propiedades de la siguiente manera:

(ID)	: Ctrl
BorderStyle	: Dashed
Text	: Test Border

3. Verifique el código ASP generado sea similar al que se muestra:

```
<asp:Label Borderstyle="dashed" Text="Test de borde" id="Ctrl"
```

```
runat = "server"/>
```

4. En el método **Page_Load** añada el siguiente código, luego ejecute la aplicación para ver su funcionamiento.:

```
protected void Page_Load(object sender, EventArgs e)
{
    Ctrl.BorderStyle = BorderStyle.Groove;
    Ctrl.Text = "Desarrollo de Aplicaciones";
}
```

b) COLORES

1. Agregue el namespace **System.Drawing**;

```
using System.Web.UI.HtmlControls;
using System.Drawing;
```

2. En el método **Page_Load** añada el siguiente código que fija el color de 3 maneras distintas al texto del control **Label**, luego ejecute la aplicación para ver su funcionamiento

```
//ARGB
int alpha = 255, red = 0, green = 255, blue = 0;
Ctrl.ForeColor = Color.FromArgb(alpha, red, green, blue);

//Color .NET
Ctrl.ForeColor = Color.Crimson;
//Color HTML
Ctrl.ForeColor = ColorTranslator.FromHtml("Blue");
```

c) FUENTE

1. En el método **Page_Load** añada el siguiente código en la parte final para cambiar las características de la Fuente. Ejecute la aplicación para ver su funcionamiento.

```
Ctrl.Font.Name = "Verdana";
Ctrl.Font.Bold = true;
Ctrl.Font.Size = FontUnit.Small;
Ctrl.Font.Size = FontUnit.Point(14);
```

d) CONTROL CHECKBOXLIST

1. Agregue al formulario un control **CheckBoxList** (ID: **chk1st**), un control **Button** (ID: **cmdOK**) y un control **Label** (ID: **lblResult**).
2. En la función **Page_Load** agregue el siguiente código:

```
if(!this.IsPostBack)
{
    chk1st.Items.Add("C");
    chk1st.Items.Add("C++");
    chk1st.Items.Add("C#");
    chk1st.Items.Add("C");
    chk1st.Items.Add("VB .NET");
    chk1st.Items.Add("Pascal");
}
```

3. En la función **cmdOK_Click** agregue el siguiente código:

```
protected void cmdOk_Click(object sender, EventArgs e)
{
    lblResult.Text="Tu escogiste: <base>";
    foreach(ListItem lstItem in chk1st.Items)
    {
        if (lstItem.Selected == true)
```

```

        lblResult.Text += "<br/>" + lstItem.Text;
    }
    lblResult.Text += "<br/>";
}

```

3

4. Ejecute la aplicación y verifique el funcionamiento

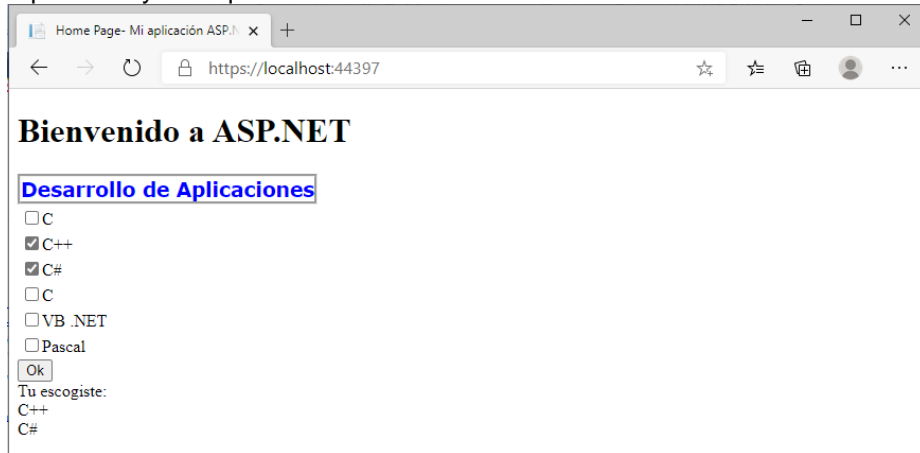


Figura N° 1: Aplicación con webform

e) CONTROL BULLETEDLIST

1. Crear una nueva aplicación web, debe grabarla con el nombre de **Tips**, asegúrese de elegir la opción correcta y eliminar el código innecesario.
2. Seleccionar el archivo Default.aspx, en la vista diseño del formulario web añada un control **BulletedList**.

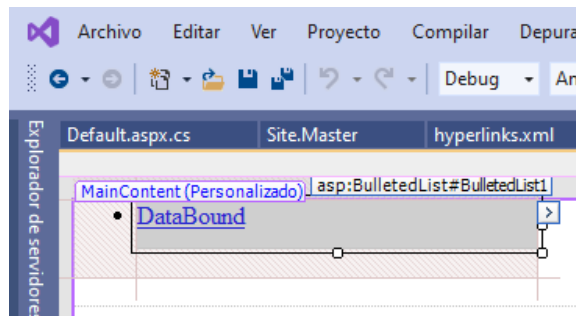


Figura N° 2: Control BulletedList insertado

3. Seleccione el control agregado y seleccione la opción para **Editar** los elementos haciendo clic sobre el botón

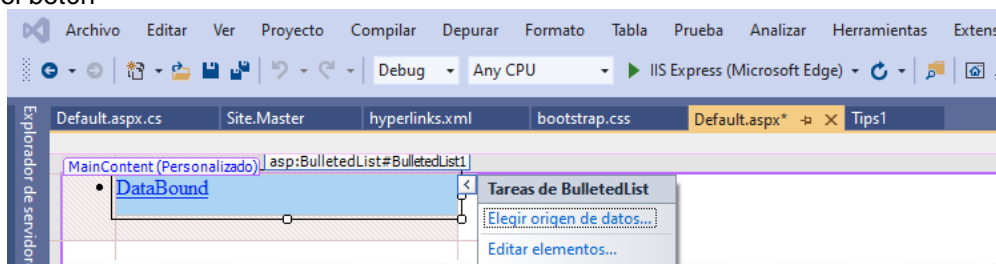


Figura N° 3: Editar Elementos del control

4. En el cuadro de diálogo Editor de la colección **ListItem**, Agregue un nuevo item, y modifique la propiedad **Text** y **Value** como se muestra en la figura.

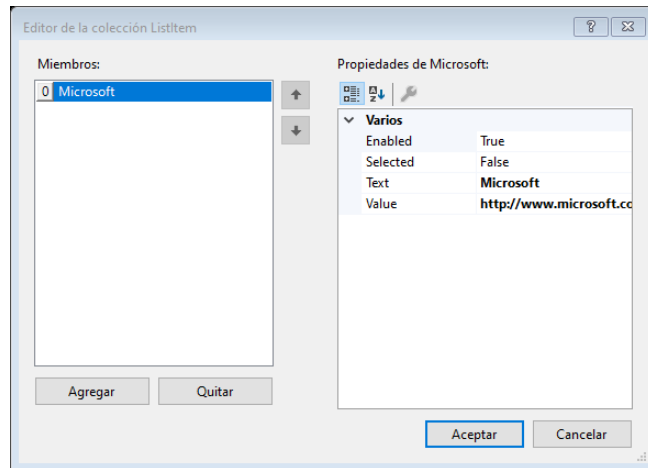
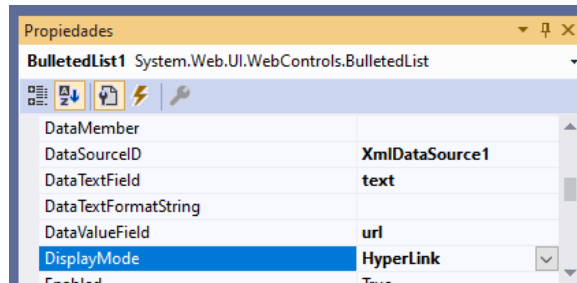



Figura N° 4: Agregar elementos al control.

- Una vez agregado el ítem, haga clic en el botón **Aceptar**. Seleccione el control **BulletedList** y modifique la propiedad **DisplayMode** seleccionando **HyperLink**. Esto permitirá que los ítems del **BulletedList** sean mostrados como hipervínculos. Ejecute la aplicación y verifique su funcionamiento.



- En el Explorador de soluciones, seleccione la aplicación **Tips**, haga clic derecho y seleccione **Agregar nuevo elemento**. De las Plantillas escoja **Archivo XML** y asígnele el nombre **hyperlinks.xml** y haga clic en el botón Agregar. En el archivo agregado escriba el siguiente código:

```
<?xml version="1.0" encoding="utf-8" ?>
<links>
  <link text="Microsoft" url="http://www.microsoft.com"/>
  <link text="MSDN" url="http://msdn.microsoft.com"/>
  <link text="ASP.net" url="http://www.asp.net"/>
</links>
```

- En la vista diseño del archivo Default.aspx, seleccione el control **BulletedList**, haga clic sobre el botón  y seleccione **Elegir origen de datos**.
- En el cuadro de diálogo Asistente para la configuración de orígenes de datos, en Seleccionar un origen de datos, escoja **Nuevo origen de datos**.
- Luego seleccionamos como tipo de origen de datos, **Archivo XML** y hacemos clic en **Aceptar**.

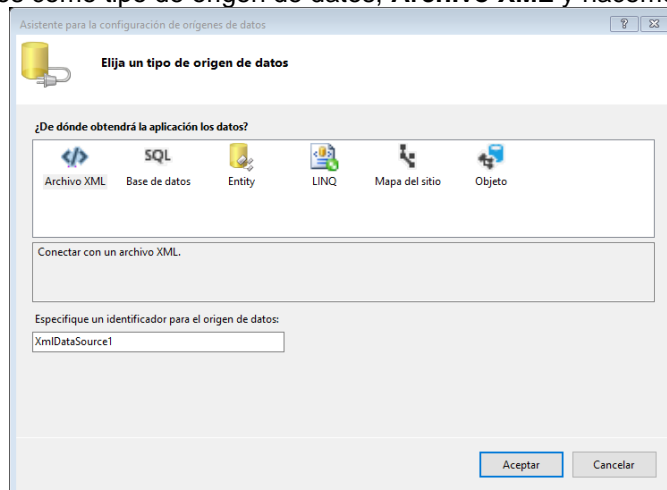


Figura N° 5: Elegir origen de datos.

10. En el cuadro de diálogo **Configurar origen de datos**, haga clic en el botón **Examinar** y seleccione el archivo **hyperlinks.xml**.

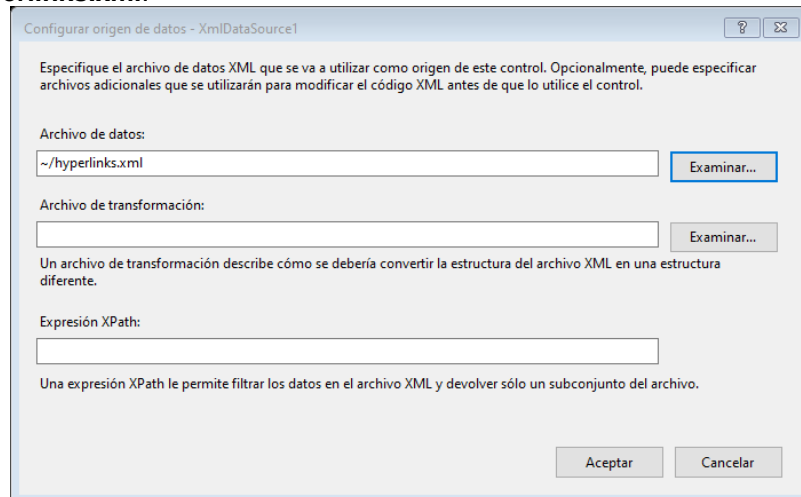
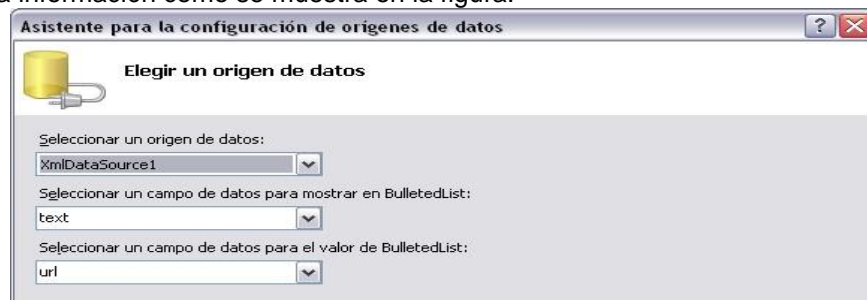


Figura N° 6: enlazar el archivo hyperlinks.xml.

11. Una vez seleccionado el archivo haga clic en Aceptar. Se debe haber agregado un control XMLDataSource.
12. Una vez seleccionado el archivo haga click en Aceptar. Se debe haber agregado un control XMLDataSource.
13. Seleccione el control BulletedList y seleccione **Elegir origen de datos**
14. Modifique la información como se muestra en la figura:



15. Ejecute la aplicación y verifique su funcionamiento.

f) CONTROL FILEUPLOAD

- En la página Default.aspx, del cuadro de herramientas, de la sección **HTML**, arrastre un Control **HorizontalRule** en el Formulario Web, justo debajo del XMLDataSource.
- Agregue en su formulario un control **FileUpload**, del cuadro de herramientas, de la sección **Estándar**. Agregue también un control **Button** y un **Hyperlink**. El diseño debería quedar como la imagen.
- En el explorador de soluciones, haga clic derecho sobre el proyecto y agregue una **Nueva Carpeta**. Dele por nombre **upload**, será la carpeta donde se guardarán los archivos subidos al servidor.
- En el evento **Click** del button, adicione el siguiente código con la ruta donde se encuentre la carpeta upload.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if(FileUpload1.HasFile)
    {
        FileUpload1.SaveAs(@"C:\Users\ucsm\source\repos\Tips1\upload\"
            + FileUpload1.FileName);
        HyperLink1.Text = FileUpload1.FileName;
        HyperLink1.NavigateUrl=@"upload/" + FileUpload1.FileName;
    }
}
```

5. Ejecute la aplicación y verifique su funcionamiento. Con el botón examinar seleccione un archivo de imagen (por ejemplo) y luego haga click en botón Upload, luego acceda al recurso subido al servidor mediante el hipervínculo.

g) CONTROL MULTIVIEW

1. Del cuadro de herramientas, de la sección *Estándar*, arrastre un control **Multiview**. Dentro del control Multiview agregado, adicione dos controles **View**. Como se muestra en la figura:

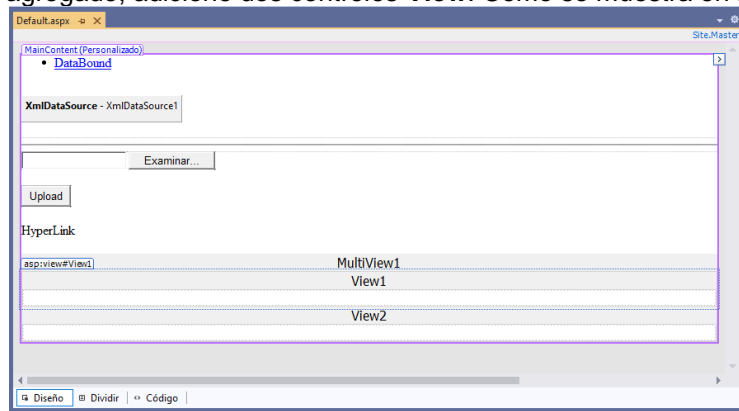


Figura N° 7: Controles View yMultiview

2. Seleccione los controles utilizados en el ejemplo del control **BulletedList**, córtelos y péguelos dentro de la Vista 1 del control MultiView. Luego seleccione los controles usados en el ejemplo del control FileUpload, córtelos y péguelos en la Vista 2 del control **MultiView**, como se aprecia:

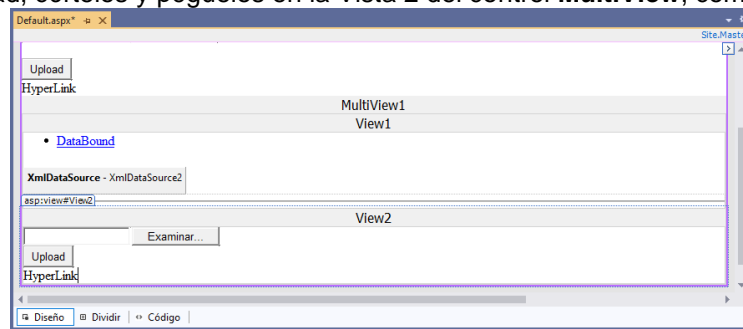


Figura N° 8: Controles configurados

3. Encima del control **Multiview**, agregue un control **RadioButtonList**. Habilite el **Autopostback** y adicione dos ítems como muestra la figura.

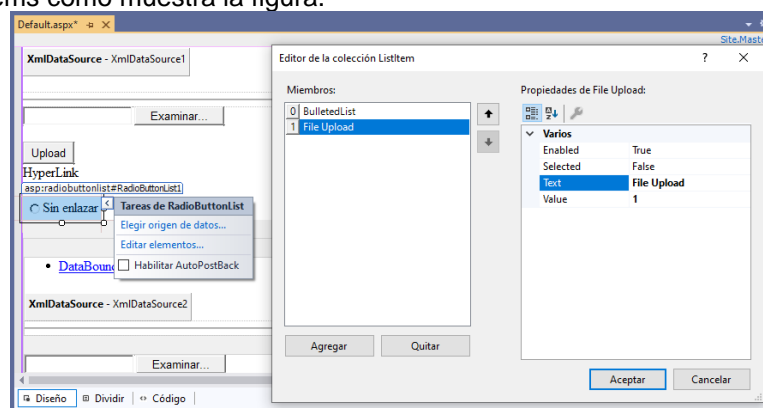
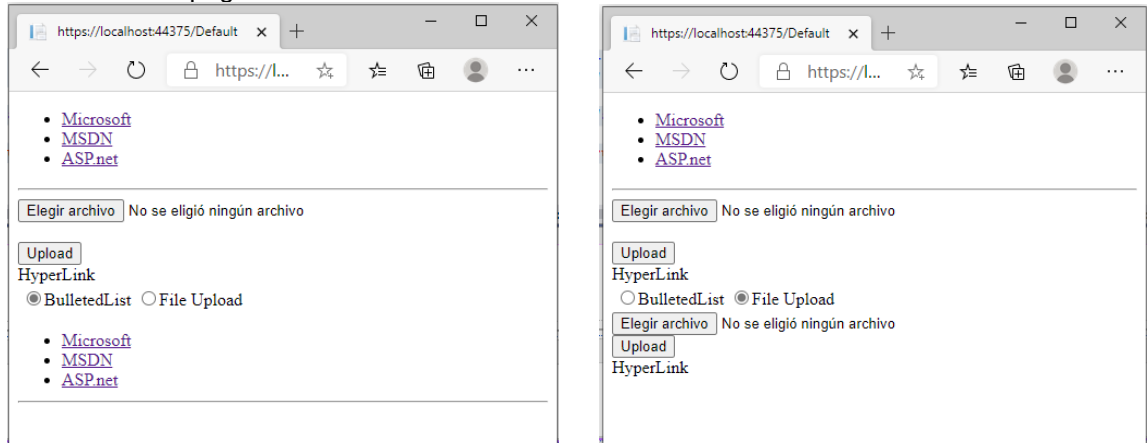


Figura N° 9: Configurar RadioButtonList.

4. Seleccione el control **RadioButtonList**, y modifique la propiedad **RepeatDirection**, con el valor de **Horizontal**.
5. Adicione el evento **SelectedIndexChanged** al RadioButtonList, y escriba el siguiente código:

```
protected void RadioButtonList1_SelectedIndexChanged(object sender,
                                                    EventArgs e)
{
    MultiView1.ActiveViewIndex =
        Int32.Parse(RadioButtonList1.SelectedValue);
}
```


6. Ejecute la aplicación y verifique el funcionamiento de la misma. Note la forma de visualizar el contenido de la página utilizando el control **Multiview**.



2. EXPERIENCIA DE PRÁCTICA N° 02: USO DE CSS

a) MENÚ VERTICAL:

Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<div id="menu">
<p><a href="http://www.google.com">Google</a></p>
<p><a href="http://www.yahoo.com">Yahoo</a></p>
<p><a href="http://www.bing.com">Bing</a></p>
<p><a href="http://www.altavista.com">Altavista</a></p>
</div>
</body>
</html>
```

Escriba el siguiente código en un archivo estilos.css:

```
#menu {
    font-family: Arial;
}

#menu p {
    margin:0px;
    padding:0px;
}

#menu a {
    display: block;
    padding: 3px;
    width: 160px;
    background-color: #f7f8e8;
    border-bottom: 1px solid #eeeeee;
    text-align:center;
}

#menu a:link, #menu a:visited {
    color: #ff0000;
    text-decoration: none;
}

#menu a:hover {
    background-color: #336699;
    color: #ffffff;
}
```

Cargue la página en el navegador de su preferencia.

b) MENÚ HORIZONTAL:

1. Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<ul id="menuhorizontal">
<li><a href="http://www.google.com">Google</a></li>
<li><a href="http://www.yahoo.com">Yahoo</a></li>
<li><a href="http://www.bing.com">Bing</a></li>
<li><a href="http://www.altavista.com">Altavista</a></li>
</ul>
</body>
</html>
```

Escriba el siguiente código en un archivo estilos.css:

```
#menuhorizontal {
  margin:0;
  padding:0;
  list-style-type:none;
}
#menuhorizontal a {
  width:100px;
  text-decoration:none;
  text-align:center;
  color:#ff0000;
  background-color:#f7f8e8;
  padding:3px 5px;
  border-right:1px solid blue;
  display:block;
}

#menuhorizontal li {
  float:left;
}

#menuhorizontal a:hover {
  background-color:#336699;
}
```

Cargue la página en el navegador de su preferencia.

c) ESTILOS A UN FORMULARIO:

1. Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<div id="contenedorform">
<form method="post" action="#">
<label>Ingrese nombre:</label>
<input type="text" name="nombre" size="30">
<br>
<label>Ingrese mail:</label>
<input type="text" name="mail" size="45">
<br>
<label>Comentarios:</label>
<textarea name="comentarios" cols="30" rows="5"></textarea>
<br>
<input class="botonsubmit" type="submit" value="confirmar">
</form>
</div>
</body>
</html>
```

Escriba el siguiente código en un archivo estilos.css:

```
#contenedorform {
  width:500px;
  margin-left:20px;
  margin-top:10px;
  background-color:#ffe;
  border:1px solid #CCC;
  padding:10px 0 10px 0;
}

#contenedorform form label {
  width:120px;
  float:left;
  font-family:verdana;
  font-size:14px;
}
.botonsubmit {
  color:#f00;
  background-color:#bbb;
  border: 1px solid #fff;
}
```

Cargue la página.

d) DISPOSICIÓN DE 3 COLUMNAS, CABECERA Y PIE.

1. Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<div id="contenedor">
  <div id="cabecera">
    <h1>Aca el título de la página</h1>
  </div>
  <div id="columna1">
    <p>columna1. columna1. columna1. columna1. columna1. columna1.
columna1. columna1. columna1. columna1. columna1. columna1. columna1.
columna1. columna1. columna1. columna1. columna1. columna1. columna1.
columna1. columna1.</p>
  </div>
  <div id="columna3">
    <p>columna3. columna3. columna3. columna3. columna3. columna3.
columna3. columna3. columna3. columna3. columna3. columna3. columna3.
columna3. columna3. columna3. columna3. columna3. columna3. columna3.
columna3. columna3. </p>
  </div>
  <div id="columna2">
    <h2>Título de la columna</h2>

    <p>Contenido de la columna2. Contenido de la columna2. Contenido
de la columna2. Contenido de la columna2. Contenido de la columna2.
Contenido de la columna2. Contenido de la columna2. Contenido de la
columna2. Contenido de la columna2. Contenido de la columna2. </p>
  </div>
  <div id="pie">
    Pié de página.
  </div>
</div>
</body>
</html>
```

- Escriba el siguiente código en un archivo estilos.css:

```
* {
  margin:0;
  padding:0;
}
#contenedor
{
  width:100%;
  border:1px solid #000;
  line-height:130%;
  background-color:#f2f2f2;
}
#cabecera
{
  padding:10px;
  color:#fff;
  background-color:#becdfe;
  clear:left;
}
#columna1
{
  float:left;
  width:200px;
  margin:0;
  padding:1em;
}
#columna2
{
  margin-left:210px;
  margin-right:230px;
  border-left:1px solid #aaa;
  border-right:1px solid #aaa;
  padding:1em;
}
#columna3
{
  float:right;
  width:200px;
  margin:0;
  padding:1em;
}
#pie {
  padding:10px;
  color:#fff;
  background-color:#becdfe;
  clear:left;
}
```

Cargue la página.

e) CSS3: BORDES REDONDEADOS.

1. Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners1 {
  border-radius: 25px;
  background: #8AC007;
  padding: 20px;
  width: 200px;
  height: 150px;
}
```

```
#rcorners2 {
    border-radius: 25px;
    border: 2px solid #8AC007;
    padding: 20px;
    width: 200px;
    height: 150px;
}

#rcorners3 {
    border-radius: 25px;
    background: url(paper.gif);
    background-position: left top;
    background-repeat: repeat;
    padding: 20px;
    width: 200px;
    height: 150px;
}
</style>
</head>
<body>

<p> La propiedad border-radius permite añadir esquinas redondeadas a los elementos .</p>
<p> Esquinas redondeadas para un elemento con un color de fondo específico:</p>
<p id="rcorners1">Rounded corners!</p>
<p> Esquinas redondeadas para un elemento con un borde:</p>
<p id="rcorners2">Rounded corners!</p>
<p> Esquinas redondeadas para un elemento con una de imagen de fondo:</p>
<p id="rcorners3">Rounded corners!</p>

</body>
</html>
```

Cargue la página.

f) CSS3: FONDOS CON IMÁGENES COMPLETAS.

1. Escriba el siguiente código en un archivo html:

```
<!DOCTYPE html>
<html>
<head>
<style>
html {
    background: url(img_flower.jpg) no-repeat center center fixed;
    background-size: cover;
}

body {
    color: white;
}
</style>
</head>
<body>

<h1>Full Page Background Image</h1>
<p>Descripción del producto.</p>

</body>
</html>
```

2. Cargue la página en el navegador de su preferencia.

3. EXPERIENCIA DE PRÁCTICA N° 01: USO DE SASS

a) SINTAXIS DE SASS

1. Como entendemos que al usar un preprocesador es preferible mantenerse más cerca del lenguaje CSS, en este manual usaremos siempre sintaxis SCSS.

Ahora veamos algunas cosas sobre la sintaxis de Sass. Por ejemplo, el uso de variables.

```
$color-fondos: #F55;  
h1 {  
  background-color: $color-fondos;  
}
```

En este código estás viendo cómo declaramos una variable y a continuación la usamos en un selector. Obviamente, eso no es un CSS válido, ya que no existen este tipo de declaraciones en el lenguaje.

2. Una vez compilado el código anterior, obtendrías un código estándar como el siguiente:

```
h1 {  
  background-color: #F55; }
```

Veremos más sobre variables y muchas otras utilidades de Sass en breve. Pero ahora preferimos dedicar algo de tiempo para explicarte cómo puedes usar Sass en tu proyecto.

b) CÓMO USAR SASS

1. Para usar Sass tienes dos alternativas fundamentales.

Preprocesar con alguna herramienta de interfaz gráfica, como el caso de Prepros, CodeKit o Scout-App. En principio puede ser más sencillo, ya que no requiere trabajar con la línea de comandos, pero necesitas usar un programa en concreto y no siempre puede estar disponible para ti, o no integrarse en otro flujo de trabajo que puedas tener ya asumido en tu proyecto. Además, las mejores herramientas de interfaz gráfica tienen la desventaja de ser de pago, o necesitar una licencia para desbloquear todo su poder.

Usar la línea de comandos para preprocesar. Esta es la opción preferida por la mayoría de desarrolladores. No sólo porque no requiere la compra de una licencia por el software de interfaz gráfica, sino principalmente porque la puedes integrar con todo un ecosistema de herramientas para optimizar multitud de aspectos en un sitio web. Además, está al alcance de cualquier desarrollador, ya que todos tenemos un terminal en nuestro sistema operativo y finalmente, permite personalizar completamente el comportamiento del preprocesador.

Usar herramientas de automatización. Como tercera opción es muy común también usar herramientas que permiten automatizar el flujo de trabajo frontend, compilando archivos CSS, Javascript, optimizando imágenes, etc. Nos referimos a paquetes como Gulp, Grunt o Webpack (aunque este último es más un empaquetador). Estos sistemas tienen la particularidad que sirven para cubrir todas las necesidades de trabajo con los lenguajes de la web y no solamente se usan para compilar el código Sass. Sería la opción más potente de las tres comentadas, aunque requiere mayor formación para poder usarlas. En este grupo también podríamos unir a PostCSS, aunque esta herramienta solamente nos sirve para convertir el CSS, aplicando diversos plugins, entre los que podría estar la compilación de Sass, y no entra en áreas como el Javascript.

Las anteriores alternativas pueden marear un poco si no estás acostumbrado a oír acerca de tantas tecnologías. No te preocupes demasiado porque vamos a irnos a una opción sencilla que te permita comenzar a usar Sass, sin demasiado esfuerzo ni configuración. En concreto te vamos a explicar ahora a usar Sass desde tu terminal, con las herramientas "oficiales" del propio

preprocesador (la alternativa planteada en el punto 2 anterior). Es una posibilidad al alcance de cualquier lector. Nuestro siguiente paso entonces será comenzar por instalar el preprocesador.

2. Instalar Sass: La instalación de Sass depende del sistema operativo con el que estás trabajando. Aunque todos requieren comenzar instalando el lenguaje Ruby, ya que el compilador de Sass está escrito en Ruby.

Veamos, para cada sistema, cómo hacernos con Ruby.

Windows: Existe un instalador de Ruby para Windows. <https://rubyinstaller.org/> Puedes usarlo para facilitar las cosas.

Linux: tendrás que instalar Ruby usando tu gestor de paquetes de la distribución con la que trabajes, apt-get, yum, etc.

Mac: Ruby está instalado en los sistemas Mac, por lo que no necesitarías hacer ningún paso adicional.

Una vez instalado el lenguaje Ruby tendremos el comando "gem", que instala paquetes (software) basados en este lenguaje. Así que usaremos ese comando para instalar Sass. De nuevo, puede haber diferencias entre los distintos sistemas operativos.

Windows: Tendrás que abrir un terminal, ya sea Power Shell, "cmd" o cualquiera que te guste usar. Si no usas ninguno simplemente escribe "cmd" después de pulsar el botón de inicio. Luego tendrás que lanzar el comando.

3. Compilar archivos de Sass a CSS: Ahora que tienes Sass instalado querrás compilar el código de Sass para convertirlo a CSS estándar. Para ese paso, necesitamos partir de un archivo con código Sass.

Antes en este artículo hemos visto cómo podría ser un pedazo de código en Sass, con sintaxis SCSS. Puedes copiar ese código y guardarlo en un archivo en tu ordenador. Usa cualquier carpeta para hacer esta primera prueba y guarda el archivo con el nombre "ej.scss" (o cualquier otra cosa que te apetezca).

Ahora te debes situar con el terminal en la carpeta donde tengas el archivo que acabas de crear con el código SCSS. Entonces lanzarás el comando para compilar, de esta manera:

```
sass ej.scss ej-compilado.css
```

Como puedes observar, al comando "sass" le indicamos el archivo origen (con código SCSS) y el archivo de destino, donde nos colocará el código CSS estándar.

Podrás observar también, una vez ejecutado el comando, que nos crea el archivo "ej-compilado.sass". Además que ha creado un archivo ej-compilado.css.map, que sirve para que el navegador pueda corresponder (en las herramientas de desarrollo) el código compilado con la referencia donde está el código original, permitiendo ayudarte a la hora de depurar. Hablaremos de mapas más adelante, de modo que no necesitas preocuparte de ese segundo archivo ".map".

4. Compilar Sass de manera automática con un "watcher": Habrás observado lo rápido que se ha compilado el archivo Sass para convertirse en código CSS estándar. Pero sin embargo, ejecutar ese comando para compilación cada vez que cambia tu código es un poco tedioso, porque te obliga ir a la consola y lanzar nuevamente el comando con cada pequeña edición que hagas.

Lo que generalmente querrás es tener un vigilante "watcher" que se encargue de compilar automáticamente el fichero cada vez que guardes el archivo original. Así, tu código compilado estará siempre fresco y podrás ahorrar unos segundos preciosos con cada compilación. Al cabo del día significará mucho tiempo y optimizará tu trabajo, haciéndote más feliz.

Paralelamente, el watch nos permite observar un archivo en concreto, o todos los archivos de una carpeta.

5. Vigilar las modificaciones de un archivo en concreto: Lo consigues con la opción --watch, indicando el archivo origen y destino, igual que hacíamos antes.

```
sass --watch origen.scss destino.css
```


Vigilar todos los archivos de una carpeta

En este caso puedes decirle a Sass la carpeta de origen y la carpeta de destino, donde colocará el código CSS estándar. Separas la carpeta origen de la carpeta de destino con el carácter dos puntos ":".

```
sass --watch carpetaorigen:carpetadestino
```

Ten en cuenta que esas carpetas existirán en tu proyecto y que el comando lo debes invocar desde la raíz del proyecto.

Ya los nombres de las carpetas podrán variar según tus preferencias. Para comenzar, si no tienes ningún requisito en este sentido, te sugerimos crear dos carpetas. Una llamada "sass" donde colocarás tu código fuente, con archivos de extensión ".scss". Otra llamada "css", donde se colocará el código compilado.

Ahora, lanzarás el watcher con un comando como este (Situado en la raíz del proyecto, donde has visto que está el archivo index.html).

```
sass --watch sass:css
```

Una vez que crees código SCSS en la carpeta "sass", se irán generando los archivos compilados en la carpeta de destino. Por ejemplo, así te quedaría la estructura de archivos y carpetas al crear tu código Sass en la ruta "sass/estilo.scss".

cuando abordemos la organización del código mediante diversos archivos fuente de SCSS, explicaremos cómo puedes evitar que se compilen todos los archivos de Sass y sólo se compile uno de ellos, el raíz. Te adelantamos que puedes evitar que el watcher los procese simplemente nombrando esos archivos con un guión bajo en su inicio. Algo como "_variables.css".

6. Usar los archivos de código generado: Como hemos dicho, el código SCSS no es compatible con los navegadores, por lo que tenemos que usar el código compilado. Esto quiere decir que, en tu index.html, así como en cualquier otro archivo HTML de tu proyecto, tendrás que enlazar con el código generado y nunca con el código fuente Sass.

Por tanto, en el index.html que has visto en las anteriores imágenes, tendríamos que enlazar con la hoja de estilos compilada, de esta manera:

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

Si has hecho todo bien, deberías ver como te pilla los estilos generados y funciona todo perfectamente.

Con esto ya has aprendido a dar los primeros pasos con Sass y has hecho lo más difícil, crear un entorno de desarrollo amigable, que permita la compilación automática de los archivos Sass cada vez que vamos introduciendo cambios.

Ahora nos queda aprender bien el lenguaje y sacarle todo el partido, manteniendo las buenas prácticas. En los próximos artículos del manual iremos abordando todos los puntos que debes saber, así como otras aproximaciones para compilar Sass que te pueden venir bien en otros casos.

V

EJERCICIOS PROPUESTOS

1. Desarrollar una aplicación que nos permita cargar información en la página Web la información sobre: el producto, el cliente y el proveedor.
2. Agregar al formulario los controles web necesarios para que sea eficiente la consulta de información.
3. Aplicar características de presentación a través de CSS.
4. Agregar características de presentación a través de SASS.

CUESTIONARIO

1. ¿Qué es un Formulario web?
2. ¿Qué son los Control Web en ASP.NET?
3. ¿Cuáles son las clases de controles web principales?
4. ¿Qué papel cumple la propiedad Page?IsPostBack?
5. ¿Cómo funciona el control MultiView?
6. ¿Qué son controles web intrínsecos?
7. ¿Qué son hojas de estilo?
8. ¿Qué características maneja CSS?
9. ¿Cuáles son las principales propiedades manejadas por CSS?
10. ¿Cuáles son las formas de agregar características CSS a una página Web?
11. ¿Cómo se incrusta características CSS en una página Web?
12. ¿Qué es SASS?
13. ¿Cuáles son las características de SASS?
14. ¿Cuáles son las principales ventajas de SASS frente a CSS?
15. ¿Cómo se configuran las características de SASS?
16. ¿Cómo se incluye las características SASS a una página Web?

BIBLIOGRAFIA Y REFERENCIAS

- Microsoft Official Course, Developing Web Applications using Microsoft Visual Studio 2010, Microsoft Cooperation, 2010
- Sánchez Flores, “Desarrollado aplicaciones con Visual C# NET 2008”. Ed. Macro, Perú, 2008
- Joseph Albarari y Ben Albarari, C# 4.0 in a Nutshell, O'Reilly Media., 2010
- Matthew MacDonald, “Beginning ASP.NET 2.0 in C# 2005”, Ed. Apress, USA, 2006.