Argentina $8,90 (recargo al interior $0,20) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

Desarrollador .net

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

14

XML
Fundamentos - Sintaxis
Esquemas y validaciones - XPath

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

ADO.NET avanzado
DataSet - System.Data
La cadena de conexión

ISBN 978-987-1347-43-8
00014
9 789871 347438

RedUSERS

COMUNIDAD DE TECNOLOGIA

¡SUMATE!

EL SITIO Nº1 DE TECNOLOGIA

Noticias al instante // Entrevistas y coberturas exclusivas // Análisis y opinión de los máximos referentes // Reviews de productos // Trucos para mejorar la productividad // Registrate, participa, y comparte tus opiniones

SUSCRIBITE

SIN CARGO A CUALQUIERA DE NUESTROS **NEWSLETTERS** Y RECIBÍ EN TU CORREO ELECTRÓNICO TODA LA INFORMACIÓN DEL UNIVERSO TECNOLÓGICO ACTUALIZADA **AL INSTANTE**

INGRESÁ A
redusers.com/suscribirse-al-newsletter
¡Y REGISTRATE YA!

Foros · Encuestas · Tutoriales · Agenda de eventos · Videos · ¡Y mucho más!

redusers.com

Seguinos en:
www.facebook.com/redusers
www.twitter.com/redusers
www.youtube.com/redusersvideos

LA ESTRELLA 2

Segundo repaso

A continuación, haremos un repaso general de los conceptos y las herramientas vistos en la Estrella número 2 del curso de programación .NET.

En esta sección, analizaremos aquellos conceptos que se evalúan en la Estrella 2 del examen de certificación de Microsoft. En primer lugar, haremos una síntesis de los más importantes. Luego, repasaremos algunas preguntas del examen y haremos algunos ejercicios prácticos que nos servirán para fijar los conocimientos aprendidos en esta etapa.

Desarrollo de aplicaciones Web

- Desarrollar aplicaciones Web o sobre Internet consiste en realizar programas ubicados en uno o más servidores Web. De este modo, podremos acceder a ellos en forma directa o a través de un nombre de usuario y una contraseña desde un navegador de Internet, sea cual fuera éste y sea cual fuera el sistema operativo, ya que la mayor parte de lo que hace funcional al programa está alojado en servidores Web.
- Las distintas tecnologías que existen para el desarrollo de estos programas basados en Web (ASP, PHP, Java, JavaScript, ColdFusion) se combinan con otros servicios que permiten acceder, por ejemplo, a bases de datos, para desarrollar un sitio altamente funcional, rico en información y dinámico a la vez.

Esquema de funcionamiento de una aplicación Web

- El esquema comúnmente utilizado en este proceso es el de cliente-servidor. Un usuario accede al sitio a través de un navegador de Internet, y se registra con un nombre de usuario y una contraseña para validar que está autorizado para utilizar esa aplicación. El acceso puede hacerse desde cualquier tecnología preparada para utilizar los servicios y protocolos de Internet.
- Utiliza la Red de redes como medio de transporte. Del lado del servidor hay una computadora que aloja al programa, el cual interpreta los requerimientos del usuario, interactúa con una base de datos –en caso de que el programa esté diseñado para tal fin–, y procesa la información para, al final, generar una página HTML con los resultados que serán enviados al navegador desde el cual el usuario la ha solicitado.
- El servidor es el principal protagonista en todo este proceso, ya que aloja el software y la base de datos. Es el encargado de recibir las peticiones, recolectar la información requerida y procesarla para transformarla en una página estática, la cual, por último, es enviada al usuario que la solicitó.

EXAMEN

ASP.NET, el lenguaje de programación Web

- ASP.NET es un entorno de desarrollo de aplicaciones Web que utiliza el concepto de web forms. Éstos son páginas que contienen tags cuya extensión es .aspx. Cada formulario tiene código en el que se gestiona la lógica de la página, se realiza el manejo de controles y objetos, y se modifica la página en función de lo solicitado por el usuario.
- Los formularios representan la interfaz de interacción con el usuario. El código que alojan estas páginas está dentro de ellas mezclado con el HTML propio de toda página y, en otras ocasiones, se encuentra separado en un archivo que contiene código específico. Ésta es la manera correcta de que la interfaz de usuario quede separada de la lógica de la aplicación.

Controles y tipos de controles

- El modelo de desarrollo de ASP.NET nos permite encapsular la funcionalidad de la aplicación en los controles. Éstos son piezas fundamentales para el desarrollo de la interfaz de usuario que se presenta en el navegador. Su apariencia es la misma que la de los controles propios de los entornos de desarrollo Visual Basic .NET y C#.
- Existen dos tipos de controles, los de usuario (*User Controls*) y los Web personalizados (*Web Custom Controls*). Los de usuario contienen HTML y código, y los Web personalizados son componentes compilados que se ejecutan en el servidor, encapsulando la interfaz de usuario y la funcionalidad dentro de paquetes reutilizables.

Librerías de clase y formularios Web

- Una librería de clase es un conjunto de clases utilizadas para un determinado fin. Las librerías que son distribuidas con el entorno .NET son extensas y poseen cientos de clases agrupadas en múltiples assemblies, cada uno de los cuales se empaqueta en un archivo DLL. La librería de .NET se conoce como *.NET Class Library*. Los desarrolladores que utilizan esta tecnología para programar encontrarán, dentro del framework .NET, clases diversas como: manejo de archivos XML, recursos de la computadora, control de interfaces, seguridad, controles, acceso a bases de datos, etc.
- Se conoce como web forms al modelo de programación utilizado en el servidor para generar páginas Web en forma dinámica. El marco de trabajo de los formularios Web ASP.NET ofrece las siguientes características:

  - Brinda capacidad para crear y utilizar controles de la interfaz de usuario que puedan ser reutilizados, encapsulando funcionalidades en común. Esto permite a los programadores reducir la cantidad de código que hay que escribir en una página.
  - Permite a todo programador estructurar la lógica de la página en forma ordenada y clara.
  - Permite que las herramientas de desarrollo nos proporcionen un fuerte soporte de diseño visual, para así poder diseñar una interfaz amigable con el usuario final.

Controles Web

- Son componentes que se ejecutan en el servidor. La mayoría se usa para encapsular porciones de la interfaz de usuario.
- Todo control posee propiedades y métodos para personalizarlo según las necesidades de nuestra aplicación, y para

programarlo con el fin de que interactúe con el usuario de forma óptima.

- Los controles notifican los eventos que lo componen mediante un PostBack, que se conoce como una nueva solicitud a la página. Mantienen su estado, definido por el valor en la propiedad ViewState. Éste es un mecanismo automático que mantiene el valor de los controles entre cada PostBack.
- Hay dos tipos de controles Web: los de servidor HTML y los de servidor Web. Los primeros son etiquetas HTML con un agregado de dos atributos: ID y runat=server. Cualquier control HTML puede convertirse en uno Web, simplemente, cambiando el valor de estas etiquetas.
- Algunos de los controles que podemos utilizar en el desarrollo de aplicaciones Web son: Label, Literal, Textbox, Hiperlink, Button, LinkButton, ImageButton, DropDownList, ListBox, CheckBox, RadioButtonList y CheckBoxList. Estos controles son comunes a cualquier aplicación Web o de escritorio. Nos permiten interactuar con los aspectos básicos de cualquier programa, como: visualizar textos, escribir textos, desplegar opciones, seleccionar entre distintas opciones, botón de interacción con el usuario, vínculos hacia otros formularios o páginas que componen el programa, y más.

Controles ricos

- Los controles ricos o *Rich Controls* poseen un comportamiento más profundo o avanzado respecto de los estándar de ASP.NET, lo que permite brindar una experiencia de usuario superior. La lógica del funcionamiento de estos controles está encapsulada, y es posible acceder a sus propiedades tanto por código como a través del diseñador visual. Se dice que son "ricos" en referencia a la riqueza dinámica o interactividad que poseen, característica que los diferencia del resto de los controles estáticos.
- Los controles ricos más comunes son: Control Calendar y Control AdRotator. El primero brinda interactividad con un pequeño calendario generado dinámicamente por el servidor. Como programadores, podemos dispones de propiedades tales como: día, mes, año, semana del año, número de día del año, entre otras opciones. Éstas pueden configurarse en forma automática de acuerdo con la región internacional establecida en el servidor, con la posibilidad de hacerlo a través de distintos formatos y disponiendo de métodos que permitan interactuar dinámicamente con todo lo relacionado a las fechas calendario. AdRotator es un control destinado a la mezcla en forma automática de banners informativos o publicitarios dentro de una página Web. Esto permite generar un espacio de publicidad en forma dinámica en una página.

Configuración de ASP.NET

- ASP.NET dispone de dos archivos de configuración principales, que influyen en el modo de ejecución de la aplicación.

  - web.config: Contiene la configuración principal de la aplicación Web.
  - machine.config: Incluye parámetros globales de configuración para el motor ASP.NET. Estos parámetros, en su carácter de globales, afectan a cualquiera de las aplicaciones Web que se ejecuten en el servidor de esa misma máquina.

- En determinadas oportunidades, es posible que ciertos parámetros de configuración coincidan en ambos archivos. De ser así, toda configuración

similar que se encuentre en el archivo web.config tendrá prioridad por sobre machine.config.

Autenticación

- Dentro del desarrollo de toda aplicación que maneje datos o información importante nos encontraremos con la necesidad de validar al usuario que accederá a ella, por una cuestión de seguridad.
- La autenticación es el acto de determinar que algo es verdadero o real. La manera más común utilizada en los sistemas de computación es la del nombre de usuario y contraseña.
- ASP.NET dispone de diferentes opciones de validación de usuario: por formulario, integrada de Windows y mediante Passport. La selección del tipo de autenticación que tendrá nuestra aplicación se configura a través del archivo web.config.

Mantener el estado

- Las aplicaciones ASP.NET necesitan el protocolo HTTP para comunicar el lado del cliente con el del servidor. Este protocolo es, habitualmente, desconectado, por lo cual el cliente se conecta al servidor, pide el recurso o la información que necesita, y el servidor la procesa y entrega, para luego desconectarse.
- Cuando el servidor procesa el requerimiento del usuario, genera un espacio en memoria para colocar los objetos y las variables en este espacio. Al entregar el resultado al cliente, el espacio de memoria es liberado, y entonces se destruyen todos los objetos creados.
- Existen dos maneras de mantener el estado de la aplicación: del lado del servidor y del lado del cliente. En la primera, interactúa la memoria del servidor, que permite almacenar el estado de la aplicación, de la sesión y la caché del programa. En la segunda opción, se trabaja del lado del cliente, ya sea en su disco rígido o en el texto de la página que viaja hasta el browser.

Páginas maestras

- ASP.NET 2 incluye el concepto de páginas maestras, que nos permiten crear plantillas en las que definimos el contenido común a todas las páginas de una manera sencilla y centralizada. Luego, cada página que utilice una Master Page heredará el contenido de ella y podrá además agregar sus propios elementos visuales.
- En las versiones anteriores de ASP esto no era posible. Cada una de las páginas que debía contener una información en común tenía que hacerse a mano, y el hecho de realizar un cambio determinado en una página que afectaba al resto implicaba modificar todos los web forms de la aplicación.

Temas y skins

- Los temas se encargan de agrupar skins, hojas de estilo, imágenes y otros recursos ubicados en subcarpetas, en una carpeta especial del sitio llamada APP_Themes.
- Los temas nos permiten obtener una variante de configuraciones visuales para un mismo sitio, y así cambiar la estética de la aplicación sin necesidad de modificar nada en los formularios.
- Las skins son simples archivos con extensión .skin que contienen definiciones sobre las propiedades de los controles ASP.NET del lado del servidor. Estos archivos contienen código similar al de las páginas .ASPX.

Navegación por el sitio

- La característica principal en todo sitio Web, sea cual fuera su finalidad, es contar con un sistema de navegación entre el usuario y las distintas secciones o pantallas que

lo componen. El control SiteMapPath nos permite seleccionar de manera automática la jerarquía de navegación por los distintos niveles del sitio, de modo que el usuario puede conocer en qué nivel se encuentra, estructurándolo como un árbol jerárquico. Para hacerlo, es conveniente usar un archivo XML que nos permita configurar el control SiteMapPath.

Acceso a datos

- Data Binding es una manera de establecer un enlace a datos para hacer referencia a la capacidad de ciertos controles de ASP.NET de asociar alguna de sus propiedades con un origen de datos, como el campo de una tabla de un DataSet u objeto. El enlace a datos nos permite simplificar gran parte del trabajo a la hora de mostrar información desde una base de datos en las páginas del sitio Web.
- La propiedad más importante y principal de Data Binding es establecer el DataSource, que equivale al origen de datos, y luego llamar al método DataBind del control. Este último método es el encargado de establecer el enlace entre los datos y el control que los mostrará en la página Web.

Windows Forms

- Un Form o formulario Windows es el componente contenedor donde se colocan los elementos gráficos que conforman la interfaz de usuario, conocida como ventana de la aplicación. Difiere de las aplicaciones Web, en las que el contenido es colocado en páginas de navegación Web.

Aplicaciones SDI y MDI

- Las aplicaciones Windows SDI están compuestas por uno o varios formularios Windows que se abren en forma independiente y pueden ubicarse en cualquier parte de la ventana. Poseen un formulario Windows principal, que oficia de contenedor o ventana madre, y uno o varios formularios Windows dentro de ésta, denominados formularios hijos.

Controles de diseño de interfaz en .NET

- .NET dispone no sólo de los controles incluidos en su entorno de desarrollo. También podemos incorporar controles de terceras partes para expandir aún más su funcionalidad, como ocurría con los ActiveX de versiones anteriores de Visual Basic y Visual C++.
- Si disponemos de una aplicación que usa un control ActiveX desarrollado por nosotros para un fin determinado, podemos incluirlo en cualquier aplicación hecha en Visual Basic .NET y C#, aunque haya sido pensado para versiones anteriores de estos lenguajes de programación.

Ciclo de vida de un formulario Windows

- Los componentes, variables, funciones y métodos incluidos dentro de un formulario Windows inician su ciclo de vida cuando éste es invocado o cargado en memoria. Si declaramos una variable dentro de un formulario Windows y le asignamos un valor durante el uso del formulario, este valor estará sólo disponible para el formulario, y al momento de cerrarlo, se perderá.
- La propiedad Cancel del parámetro del evento **FormClosing** permite cancelar el cierre de un Formulario, o preguntarle al usuario si desea proseguir o no con esa acción. Esto es útil para no perder datos que, sin querer, no grabamos antes del cierre del formulario.

EXAMEN

Preguntas de ejemplo

1 | ¿Qué es el desarrollo de aplicaciones Web?

A. La creación de un sitio Web con páginas HTML, textos e imágenes.
B. La creación de un sitio Web personal en un subdominio de un proveedor de Internet.
C. La creación de aplicaciones que residen en un servidor Web.

Respuesta correcta C: Estas aplicaciones utilizan Internet como lugar de alojamiento, para estar disponibles desde cualquier parte del mundo, aprovechando la expansión y la disponibilidad de Internet a nivel global.

2 | ¿Cuáles de las siguientes etiquetas o tags conforman una página Web?

A. <if>, <switch>, <Select Case>
B. , <head> </head>, <td> </td>
C. <data>, <xml>, <http>

Respuesta correcta B: Las etiquetas que conforman una página HTML deben encerrar el texto al cual le van a aplicar un formato. Dichas etiquetas deben tener un inicio entre <> y un final para marcar hasta dónde aplicar el formato que les corresponde. Al final de la etiqueta se debe incluir una barra invertida entre </>.

3 | ¿Cuál es la diferencia entre HTML y XHTML?

A. XHTML es el mismo lenguaje HTML 4.0, pero enriquecido con XML.
B. HTML y XHTML son lo mismo, con la diferencia de que XHTML es propiedad de una empresa, y HTML es un estándar que puede ser aplicado por cualquier persona o empresa.
C. XHTML incluye animaciones extra dentro de una página HTML, hecho que mejora su estética visual.

Respuesta correcta A: El lenguaje XHTML es una extensión del HTML, que utiliza documentos basados en XML para presentar contenido dentro de las páginas Web. XML es un formato de documento en el cual es muy fácil agregar tags y atributos. XHTML sirve para operar con distintos agentes, no sólo con navegadores Web.

4 | ¿Qué es ASP.NET?

A. Es un lenguaje de programación de aplicaciones Web.
B. Es un entorno de desarrollo de aplicaciones Web.
C. Es una herramienta de programación complementaria a HTML para desarrollar programas sobre Internet.

Respuesta correcta B: ASP.NET es un entorno de desarrollo de aplicaciones Web, basado en la tecnología .NET, que agrupa los lenguajes de programación de esta misma, basado en el concepto de formularios Web. Este concepto permite separar la interfaz gráfica de usuario, de la lógica de la aplicación. ASP.NET permite incluir, dentro del desarrollo Web, diversos lenguajes de programación, como JavaScript, Visual Basic .NET y C#, entre otros.

5 | ¿Qué son los ciclos de vida de una página?

A. Es el estado que ocurre en la página desde que carga hasta que termina de hacerlo, cuando el usuario se dirige hacia otra página del sitio.
B. El ciclo de vida de la página ocurre al iniciar su alojamiento en el servidor Web, hasta que ésta es borrada o movida hacia otro servidor.
C. Son los distintos estados por los que pasa la página desde el momento en que es solicitada por un cliente.

Respuesta correcta C: La página, desde que es invocada desde el servidor, pasa por distintos estados o pasos de procesamiento. Algunos de ellos son: load o carga, rendering, postback, unload, init o inicialización de página, request y response. Cada uno de ellos cumple un papel importante, al disparar distintos eventos. Dentro de cada uno también podemos escribir diferentes eventos que permitan actuar en consecuencia.

6 | ¿Qué son y cómo se utilizan los controles Web?

A. Son componentes, tales como botones, cajas de texto, menús desplegables, grillas de datos, etcétera, que se utilizan en las páginas Web para generar interacción visual entre la aplicación y el usuario. Dichos controles se incluyen como componentes en la barra de herramientas del entorno de programación ASP.NET.
B. Son componentes de Windows conocidos como ActiveX. Se utilizan comúnmente en las aplicaciones de escritorio de Windows. Para utilizarlos en desarrollo Web, hay que programarlos en lenguaje Java desde cero.

C. Son librerías incluidas dentro de la plataforma .NET, las cuales están disponibles para los lenguajes de programación C# y Visual Basic .NET. Para utilizarlas en ASP.NET hay que distribuir el framework .NET junto al sitio de nuestra aplicación Web.

Respuesta correcta A: Los controles son componentes que se ejecutan en el servidor. La mayoría de ellos se utiliza para encapsular porciones de interfaz de usuario. Se usan dentro de la página Web, ya sea escribiendo la etiqueta necesaria para que el control aparezca o arrastrándolo desde la barra de herramientas del entorno de programación Visual Developer Studio 2005 Express. Los controles poseen propiedades y métodos, en los cuales se puede incluir código para que se realicen distintos procesos y, así, enriquecer aún más el funcionamiento de nuestra aplicación Web.

7 | ¿Cómo se identifica el código correspondiente dentro de una página de los controles Web?

A. Se utiliza una etiqueta indicativa del inicio y el fin del código correspondiente al control, del tipo <Ctrl> </Ctrl>.
B. Se utiliza la etiqueta del nombre de control directamente al iniciar el código.
C. Se utiliza la etiqueta o prefijo <asp:nombredeCtrl> </asp:nombredeCtrl>, tanto en el inicio como en el fin de código correspondiente al control.

Respuesta correcta C: Los controles Web permiten trabajar más flexiblemente que los propios de HTML, por tener una interfaz más rica, y tener mejor definidos sus propiedades y métodos. En su modo declarativo (representación en la vista de código HTML), todos los controles de servidor Web utilizan el prefijo asp.

8 | ¿Cuál es la diferencia entre el control Web Label y el control Literal?

A. El control Literal es similar a Label, pero sólo Literal soporta temas o skins.
B. Label y Literal son similares en su modo de uso, pero sólo Literal está preparado para utilizarse en navegadores alternativos a Internet Explorer.
C. El control Literal es similar a Label, pero sólo Label soporta conexión a base de datos.

Respuesta correcta A: El control Literal posee una propiedad llamada Mode, que se utiliza para especificar al Web Control de qué manera manejar el HTML.
Transform permite que el HTML se acomode según el navegador donde se visualiza la página. PassThrough indica que el HTML se va a renderizar tal como es, y Encode codifica el HTML utilizando HTMLEncode. Esta última opción es útil cuando deseamos mostrar por pantalla HTML en un navegador en vez de utilizarlo.

9 | ¿Qué son los controles de lista?

A. Son controles que permiten desarrollar grillas y tablas anidadas dentro de una aplicación Web.
B. Son controles que nos permiten desplegar una serie de opciones, y de las cuales podemos seleccionar una o más según nuestra conveniencia, para realizar determinado proceso u operación en la aplicación Web.
C. Son controles del tipo Button, Image y Bullet, que poseen propiedades avanzadas y métodos que enriquecen aún más su funcionamiento.

Respuesta correcta B: Cada opción que nos brindan los controles de lista está modelada o identificada como un control ListItem. Los controles de lista presentan múltiples opciones de selección, cada una de los cuales tiene una colección de controles ListItem. Este último control representa un elemento de un control de lista, contiene una propiedad Text para el texto que mostrará y una propiedad Value para determinar el valor relacionado con ese elemento.

10 | ¿Qué nos indica la propiedad SelectedValue dentro de los controles de lista?

A. Que el control está seleccionado por un proceso o método que está realizando una operación con él.
B. Que el control posee un estado de habilitado o deshabilitado, para ser mostrado o no en el proceso de Request de una página Web.
C. Nos permite obtener el valor asociado a la opción seleccionada dentro del control de lista.

Respuesta correcta C: Las distintas propiedades y métodos que poseen los controles de este tipo nos permiten llevar un control preciso y realizar distintas operaciones en base a ellos dentro de las aplicaciones Web. SelectedValue se combina con diversas propiedades y métodos, como SelectedIndexChanged y SelectedItem, comunes a todos los controles de lista.

EXAMEN

Ejercicios con ASP.NET

Como siempre recomendamos, el mejor aprendizaje se realiza sobre la práctica. Es por eso que en esta página proponemos, al igual que en el ciere de la Estrella 1, ejercitar los conocimientos aprendidos hasta ahora con respecto al lenguaje ASP.NET. Aconsejamos que los ejercicios a continuación sean realizados utilizando tanto el lenguaje Visual Basic .Net como el lenguaje C# sobre la plataforma Visual Web Developer Studio.

Pautas para tener en cuenta

Todos los ejercicios pueden realizarse sobre el proyecto aquí nombrado. Para no perder nada del código que ya se encuentra funcionando de manera óptima, recomendamos copiar el proyecto o solución y guardarlo con el nombre que mejor lo identifique; por ejemplo Ejercicio 1 Pedido de Productos, Ejercicio 2 Pedido de Productos, y así sucesivamente. Ésta es una manera prolija de realizar modificaciones sobre un proyecto, y en caso de que por algún error el proyecto dejase de funcionar, podemos volver al anterior y comenzar todo nuevamente.

EJERCICIO 1: Combinar los conocimientos adquiridos en XML con el ejercicio "Pedido de Productos" realizado en la página 173, cargando los productos que queremos que aparezcan en el control ListBox. Para hacerlo, debemos pensar en armar una función que permita leer el archivo XML y recorrerlo, cargando los datos que en él se encuentran en el ListBox del Web Form.

EJERCICIO 2: Agregar al proyecto "Pedido de Productos" un botón que nos permita eliminar el o los productos que seleccionemos del ListBox, añadiendo una función que realice la eliminación del ítem en el control ListBox, cuando el botón es presionado. Sería bueno que el programa pidiera una confirmación antes de realizar la eliminación del ítem seleccionado.

EJERCICIO 3: Generar un ViewState en la página del proyecto "Pedido de Productos" del tipo Cookie, que al cargar dicha página, muestre la fecha y hora del último ingreso realizado. Para verificar su correcto funcionamiento, sería ideal cambiar la fecha de la PC en el momento de probarlo.

EJERCICIO 4: Crear una página maestra que contenga la página del proyecto "Pedido de Productos", y mover a ella el botón y la función "eliminarProductos".

EJERCICIO 5: Realizar un skin para personalizar el proyecto Pedido de Productos y mejorar su estética, e incluir un menú de navegación por el sitio del proyecto. Para hacerlo, hay que utilizar archivos XML para modificar la parte visual, y algún editor de imágenes (en el caso de que en el Web Form se hayan incluido varias imágenes). Si este último es el caso, probar solo cambiando los colores de las imágenes y guardándolas en otra subcarpeta. Para esto se debe incluir en el nombre de la imagen el color de la misma.

XML

El estándar que revolucionó la programación

7

Contenidos

En el siguiente capítulo, nos introduciremos en uno de los estándares más conocidos y poderosos, tanto de la programación Web, como así también de la de escritorio.

Temas tratados

- El origen y su uso
- La sintaxis
- Esquemas y validaciones
- Transformar un XML
- XPath

XML

Aprenderemos todas las características y nociones fundamentales para aprovechar al máximo todo el poder de XML.

Introducción

En esta primera etapa, haremos un repaso sobre el origen de este estándar, y veremos cuál es su uso actual.

- El origen y su uso
- Características
- Ventajas

Sintaxis de XML

Cuáles son las características particulares que tiene la programación en XML y cómo aprovechar todo su potencial.

- Partes que lo componen
- Entidades predefinidas
- Elementos primarios y secundarios

Trabajando con XML

Veremos también algunas cuestiones particulares para tener en cuenta al momento de trabajar con este estándar.

- Nodos de árbol
- La función main
- Elementos primarios y secundarios
- Esquemas y validaciones

Esquemas y validaciones

Aprenderemos cuáles son los pasos por seguir para mostrar un documento XML como si fuera uno HTML.

- Utilización de CSS
- Utilización de XSL
- La tecnología XPath

XPath

De qué se trata esta herramienta que utilizaremos con mucha frecuencia al trabajar con documentos XML.

- Visualización del árbol
- Jerarquía de espacios
- Clases de System.Xml.XPath

Introducción a XML

Conoceremos XML, un lenguaje de marcas que nació hace algunos años, aunque sus ancestros se remontan algunas décadas atrás.

Comencemos por conocer qué quiere decir exactamente la sigla XML. XML, en inglés, significa *eXtensible Markup Language* (lenguaje de marcas extensible), y es un lenguaje de marcado desarrollado por el W3C (*World Wide Web Consortium*).

El origen y su uso

XML no fue pensado sólo para usar sobre Internet, sino que fue creado como un estándar para compartir información estructurada, y cuenta con la posibilidad de que dicha información tenga la capacidad de describirse a sí misma. Hoy en día, XML es utilizado en bases de datos, editores de texto, planillas de cálculo, etc. Uno de los casos que ha demostrado tener mayor crecimiento es en el uso de la sindicación.

Para conocer un poco más sobre la historia de los lenguajes de marcas, no podemos dejar de mencionar que el primero de este tipo fue el GML (*General Markup Language*), creado por IBM en la década de 1970. Luego se transformó en SGML (*Standard General Markup Language*), que no es otra cosa que una estandarización del anterior. Cuando se creó Internet, junto con ella nació el lenguaje más famoso, el HTML. Después de un tiempo, con estos

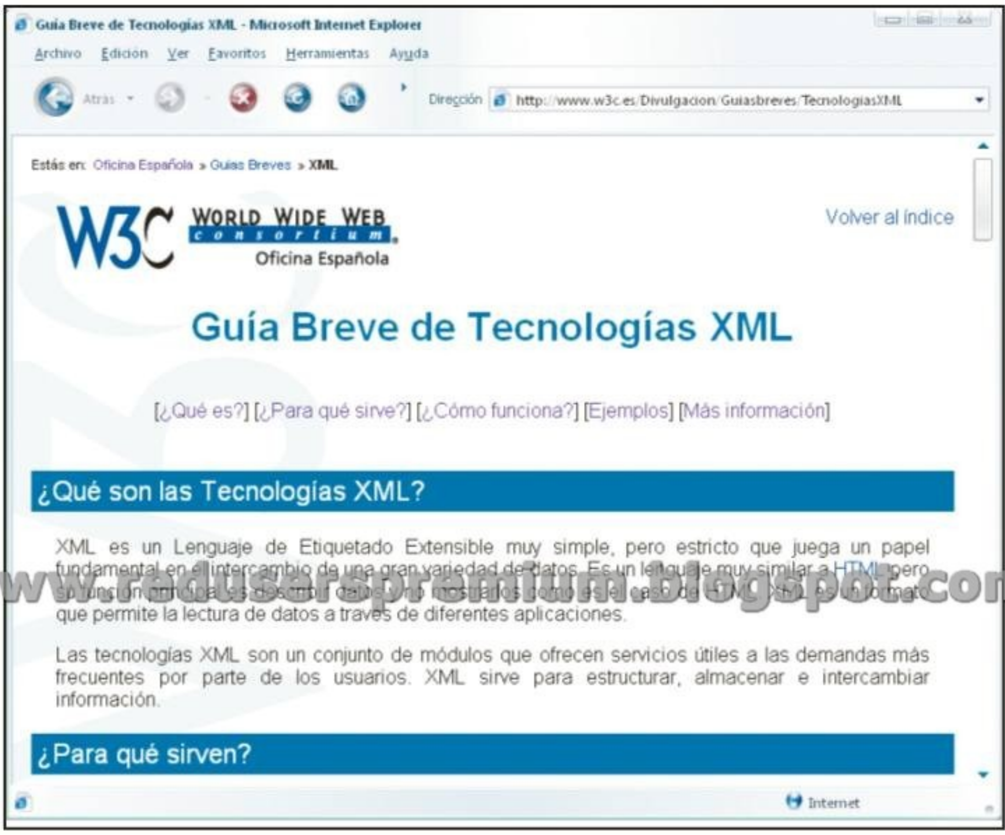


FIGURA 001 | En la edición española de la W3 Consortium encontramos una reseña acerca de esta tecnología.

XML se creó como un estándar para compartir información estructurada.

lenguajes ya en uso, y sus conocidas ventajas y desventajas, se decidió crear el lenguaje de marcas XML, que, a diferencia de sus antecesores, incluye las siguientes características:

- Permite mezclar elementos de diferentes lenguajes, lo que lo hace extensible.
- Posee analizadores simples sin lógica especial para cada lenguaje.
- No permite errores de sintaxis, lo que quiere decir que un documento XML debe estar bien escrito en cuanto a una sintaxis bien definida.

Por otro lado, la tecnología basada en XML ha ido avanzando y se incorporó una gran cantidad de utilidades con el fin de aprovechar las ventajas de un lenguaje como XML.

Sin más, veremos que XML también se ha sumado al conocido lenguaje SQL (*Structured Query Language* o lenguaje de consultas estructurado), utilizado en bases de datos. XML se ha denominado como XQL, que intenta ser lo mismo que SQL, pero con resultados XML.

También existen las llamadas hojas de estilo, con lo cual un mismo documento XML podría verse de una u otra manera según el estilo que se le aplique.

En la actualidad, XML es utilizado para definir archivos de configuración y compartir información entre aplicaciones.

Sintaxis de XML

A continuación, vamos a ver las partes de un documento XML. Éste consta de dos secciones principales: el prólogo y el cuerpo. La primera no es de carácter obligatorio pero suele usarse, al menos, para describir la versión de XML que se está empleando. Otras partes, tampoco de carácter obligatorio, son el tipo de documento, los comentarios y las instrucciones de procesamiento. La declaración de tipo de documento es la que enlaza a dicho documento con su DTD (*Document Type Definition*). El DTD también puede incluirse en esta parte del mismo documento, lo que se define como DTD interno (luego veremos más sobre este tema). La otra sección, el cuerpo del documento, es de carácter obligatorio y debe contener un elemento raíz, como veremos más adelante. Otras partes de un documento XML son los elementos que pueden estar vacíos o bien contener otros elementos o texto, o ambos.

Tabla 1 | Usos más comunes de XML

SOAP	Protocolo estándar de llamada a métodos remotos e intercambio de mensajes en XML utilizando teoría de objetos.
XML-RPC	Protocolo de llamadas de procedimientos remotos (RPC) que utiliza XML para manejar o codificar los datos.

Luego están los atributos (pueden ser parte de los elementos), que se utilizan como forma de incorporar características o propiedades a un elemento.

También podemos encontrar entidades predefinidas, que se utilizan al comienzo del documento para definir caracteres especiales. El mismo lenguaje XML incorpora algunas por defecto y que, por lo tanto, no necesitan definición, como &amp para el símbolo ampersand (&), &apos para el apóstrofo ('), &quot para las comillas ("), &lt para el símbolo de menor (<) y &gt para el de mayor (>). Por supuesto, no debemos olvidarnos de los comentarios, que se indican con < ! - - y se cierran con - - >.

A continuación, veamos en detalle un documento XML y su descripción:

```
<?xml version="1.0" encoding="utf-8" ?>
<listadoClientes>
  <!–Este es un cliente–>
  <cliente>
     <apellido>de Permentier</apellido>
     <nombre>Emilio</nombre>
     <documento tipo="dni" numero="20426385">
     </documento>
     <ciudad>Resistencia</ciudad>
     <provincia>Chaco</provincia>
     <telefono>
        <prefijo>03722</prefijo>
        <fijo>123456</fijo>
        <celular>15123456</celular>
     </telefono>
  </cliente>
  <!–Este es otro cliente–>
  <cliente>
     <apellido>Raffo</apellido>
     <nombre>Fernando</nombre>
     <documento tipo="ci" numero="11444777">
     </documento>
     <ciudad>Ciudad autónoma de Buenos Aires
     </ciudad>
     <provincia>CF</provincia>
     <telefono>
        <prefijo>011</prefijo>
        <fijo>123456</fijo>
        <celular>15123456</celular>
     </telefono>
  </cliente>
</listadoClientes>
```

XML cuenta con dos secciones principales en su estructura: el prólogo y el cuerpo del documento.

Existen varias maneras de crear un documento XML. Dentro del entorno de desarrollo Microsoft C# o Visual Basic .NET 2005 en sus versiones Express tenemos la posibilidad de generar un documento XML con resaltado de sintaxis propia de todo entorno de programación,

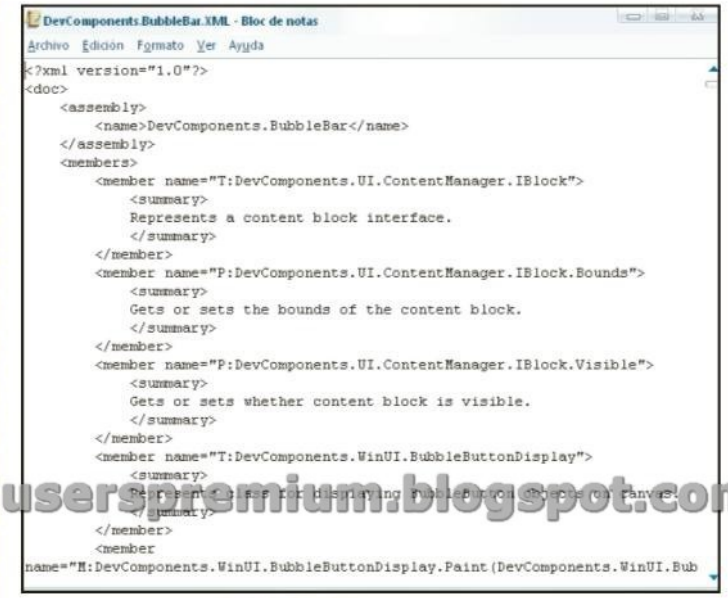


FIGURA 002 | Aquí vemos con el Bloc de notas la estructura XML perteneciente a BubbleBar, un componente gratuito para desarrollar en .NET.

para lograr una mayor comprensión de los textos que componen su código.
También podemos crear documentos XML en el Bloc de notas y verlos con cualquier navegador. Como vemos al principio, se nota claramente la versión de XML que estamos utilizando y la codificación, temas que aún no profundizaremos.
A continuación, aparece el tag <listadoClientes>, que al final del documento se cierra con el tag </listadoClientes>. Esto es lo que llamamos elemento raíz.

Luego se presentan los elementos secundarios, los cuales vamos introduciendo en forma indentada para su mejor presentación, pero que no son de carácter obligatorio. El primero de ellos es <cliente>, que si bien no tiene texto, posee elementos internos (todos los que están entre los tags de apertura y cierre, o sea, entre <cliente> y </cliente>).

Dentro del archivo, hay otro elemento denominado <apellido>, que contiene texto.

Aquí vamos a hacer hincapié en un punto muy importante propio de los documentos XML. Existen dos tipos de textos: los que pasan por el analizador de sintaxis (sintaxis de XML, por supuesto) y los que no lo hacen.
Otro aspecto de la sintaxis es que diferencia entre mayúsculas y minúsculas, por lo que si abrimos un tag "<cliente>", no podremos cerrarlo como "</CLIENTE>"; tendrá que ser iniciado y cerrado de la misma manera:

```
<cliente></cliente>
```

Los tags XML que mencionamos también reciben el nombre de etiquetas.
Otro de los elementos que encontramos en el ejemplo es <telefono>, el cual contiene otro grupo de subelementos.
También apreciamos los comentarios anteriores a cada tag "cliente". Éstos, obviamente, no pasan por el analizador de sintaxis.

Hasta aquí vimos un XML correctamente formado y de estructura bastante simple. En él

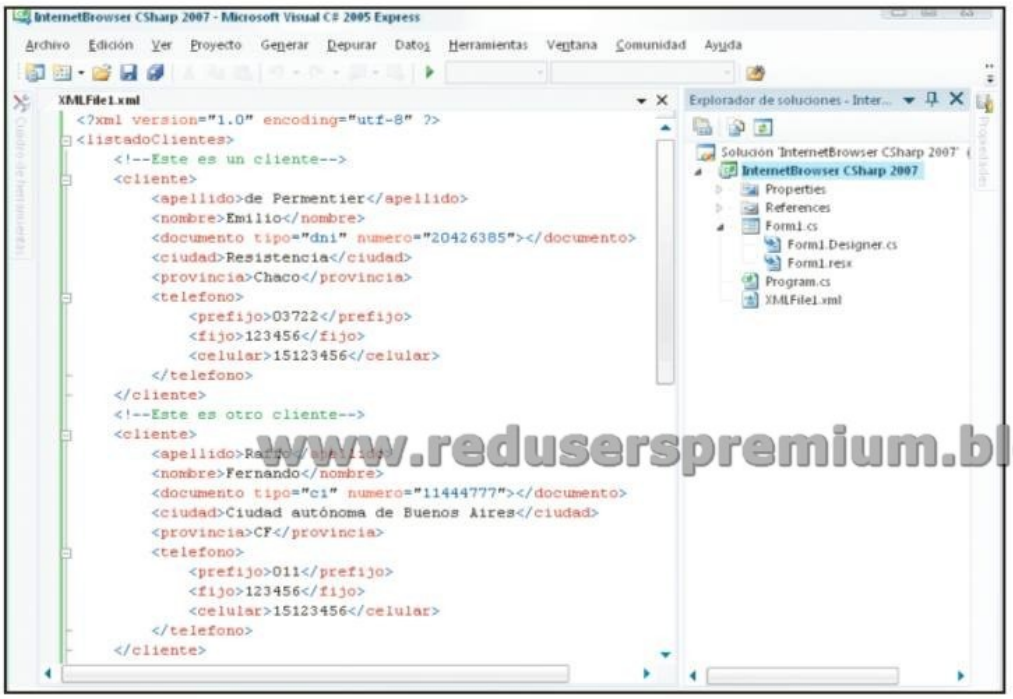


FIGURA 003 | El documento XML creado desde el entorno Microsoft C#.

podemos observar la similitud con dos registros de una base de datos, y es que los documentos XML están pensados, entre otras cosas, para compartir información.
Profundicemos ahora en el elemento <documento>, que contiene dos atributos: uno tipo y otro número, cada uno con un valor. Los atributos se utilizan para añadir información adicional a los elementos del documento. Por lo tanto, vimos un documento XML con elementos que poseen otros elementos o subelementos, los cuales, a su vez, contienen texto y, eventualmente, atributos. Cabe aclarar que podría contener las tres cosas o las diferentes combinaciones de ellas. Podemos encontrar también otras secciones; por ejemplo, las denominadas CDATA, cuya finalidad es aislar el texto para que éste no sea analizado sintácticamente. Esto se denomina texto bloqueado.

Este texto se engloba entre las cadenas <![CDATA[y]]>. Por ejemplo:

> El documento XML es recomendable trabajarlo con indentación, para lograr una mayor comprensión.

```
<doc tipo="ci" numero="11444777"><![CDATA
[datos no analizados]]></doc>
```

Esta sección no puede incluirse a nivel de elementos, pero sí entre atributos.

Sigamos conociendo un poco más cómo podemos recorrer un documento XML dentro de C# (de manera análoga en otros lenguajes). Para hacerlo, veremos que nuestro programa deberá trabajar como un analizador "de árbol" y no "sintáctico". Esto quiere decir que no veremos si el documento está bien formado (si no lo estuviera, se elevaría una excepción donde esté mal), sino que veremos cómo está compuesto para, luego, recorrerlo.

FIGURA 004 | Las distintas secciones que componen un documento XML.

Un documento XML es un objeto que contiene otros objetos con ciertas propiedades.

Aquí es donde vamos a considerar el propio documento XML como un objeto que contiene elementos, que serán tratados también como objetos, por lo que nuestro documento resulta ser un objeto que contiene otros objetos con ciertas propiedades; incluso, los comentarios también se denominan objetos.
Pero no sólo esto conforma una estructura: también podemos conocer cómo están relacionados estos objetos entre sí.
Para comenzar, veremos el objeto más básico, que es el propio documento, el objeto de texto, el objeto de elemento, el de comentario y el de atributo. Hay algunos más que por ahora no vamos a incluir para no complicar esta sección. Nuestro objeto principal (documento u objeto raíz) contiene elementos que resultan ser nodos, los cuales tendrán otros nodos, texto y/o atributos.

En la Tabla 2 se muestran los nodos del árbol y sus propiedades, para ampliar el concepto. Para trabajar con C# un documento XML debemos incluir la referencia al espacio de nombres System.Xml, con la notación que se indica en la siguiente línea:

```
using System.Xml;
```

Ahora veremos las partes que componen este espacio de nombres y cómo trabajar con ellas. Lo primero que debemos utilizar al tratar de leer un documento XML es el tipo de dato (objeto) XmlDocument.
Podemos hacerlo de la siguiente manera:

```
XmlDocument xDoc = new XmlDocument();
```

Luego, tratamos de leer nuestro documento con la siguiente sintaxis:

```
xDoc.Load("rutaamidocumento.xml");
```

Una vez que tengamos el documento cargado dentro de la instancia xDoc, podemos acceder a su información buscando su elemento raíz:

```
XmlNodeList clientes = xDoc.GetElements
ByTagName("listadoClientes");
```

En este momento estamos en condiciones de revisar nuestros clientes con una simple instrucción foreach que recorra los nodos secundarios de este elemento:

Tabla 2 | Nodos de árbol

Nodos de árbol	Propiedades
Node type	Comentario, declaración de versión de elemento, etc.
Node name	En caso de ser un elemento, sería su nombre (para el documento del ejemplo: apellido).
Node value	En caso de ser texto o nodo de comentario, sería la cadena de caracteres (no se plantean en el ejemplo).
Parent node	El nodo primario al que pertenece.
HasChildNodes	Indica si posee nodos secundarios.
Child list	Lista de nodos secundarios que posee el nodo en cuestión.

```
Foreach (XmlNode nodo in clientes.Nodes)
```

Para que esta información resulte más clara, analizaremos el documento de ejemplo y enviaremos a la consola algunos de sus datos. Para hacerlo, vamos a crear una clase en un proyecto de consola. Lo primero que hacemos es incluir los espacios de nombre necesarios:

```
using System;
using System.Xml;

namespace xmlTest
{
  class TestXml
  {
      static void Main(string[] args)
      {
      }
  }
}
```

Para trabajar con C# debemos incluir la referencia al espacio de nombres System.Xml.

Una vez creada la estructura básica de nuestra aplicación, comenzamos a incluir código dentro de la función main. Para empezar, declaramos un objeto que contendrá nuestro documento XML y lo leemos de este modo:

```
XmlDocument miDocumento = new XmlDocument();
miDocumento.Load("documento.xml");
```

Ahora que ya tenemos cargado nuestro documento, procedemos a recorrer todos los nodos "cliente" que haya. Para hacerlo, debemos cargar primero nuestro nodo raíz, en este caso denominado "clientes":

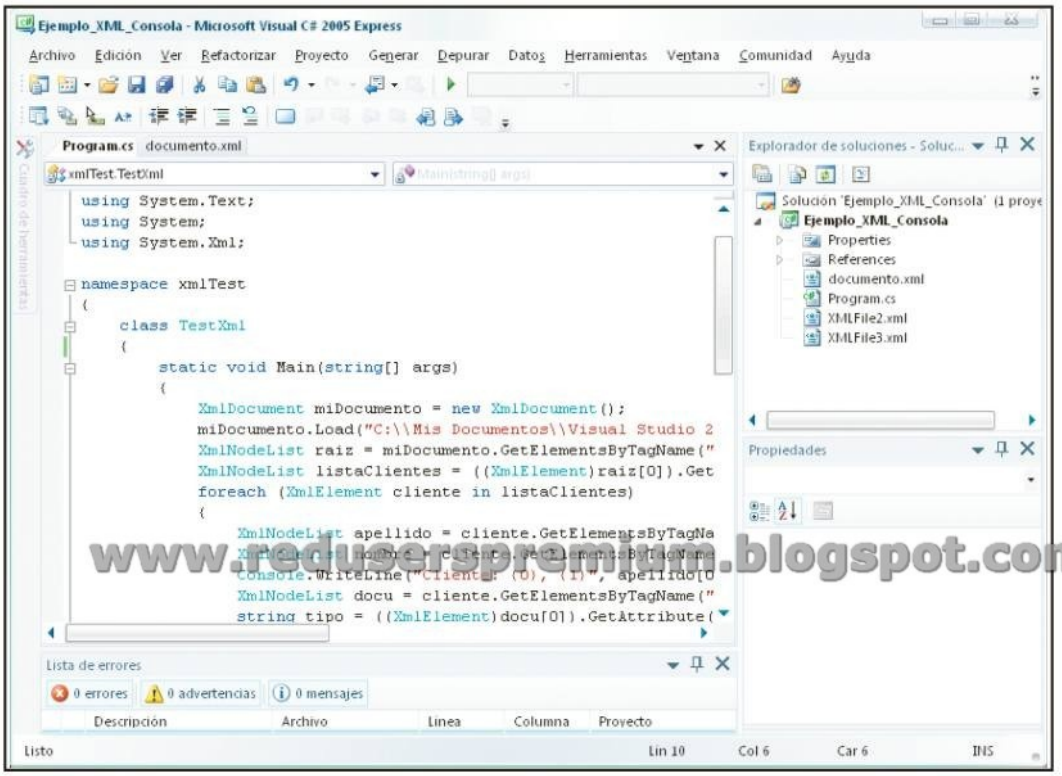


FIGURA 005 | Tanto C# como VB.NET nos permiten editar archivos XML dentro del mismo entorno de desarrollo.

```
XmlNodeList raiz = miDocumento.GetElements
ByTagName("listadoClientes");
```

Ya tenemos la información completa de la raíz, por lo que entramos en un bucle para recorrer, en este caso, todas las personas de nuestro documento XML. Primero obtenemos una lista de todos nuestros clientes, la cual vamos a recorrer:

```
XmlNodeList listaClientes = ((XmlElement)raiz
[0]).GetElementsByTagName("cliente");
```

Es decir que podemos hacerlo de a un tag por vez o bien ingresando todo el path a donde queremos ir. Ahora entramos de lleno en el bucle de clientes:

```
foreach (XmlElement cliente in listaClientes)
{
}
```

Este bucle, como bien sabemos, recorre todos nuestros clientes y nos entrega un XmlElement por cada uno, el cual debemos analizar para lograr nuestro cometido. En este ejemplo, y para ser sintéticos, sólo escribiremos por consola el apellido, el nombre, el tipo y número de documento, y el teléfono fijo antecedido del prefijo de área. Antes veremos que los nodos también son elementos, así que incluiremos el siguiente código para ir escribiendo los dos primeros valores:

```
XmlNodeList apellido = cliente.GetElements
ByTagName("apellido");
XmlNodeList nombre = cliente.GetElements
ByTagName("nombre");
```

Luego los enviamos a la consola:

```
Console.WriteLine("Cliente: {0}, {1}",
apellido[0].InnerText, nombre[0].
InnerText);
```

Continuamos mostrando el documento de esa persona, que, como podemos observar en nuestro XML, figura dentro de un mismo tag pero en forma de atributos, por lo que

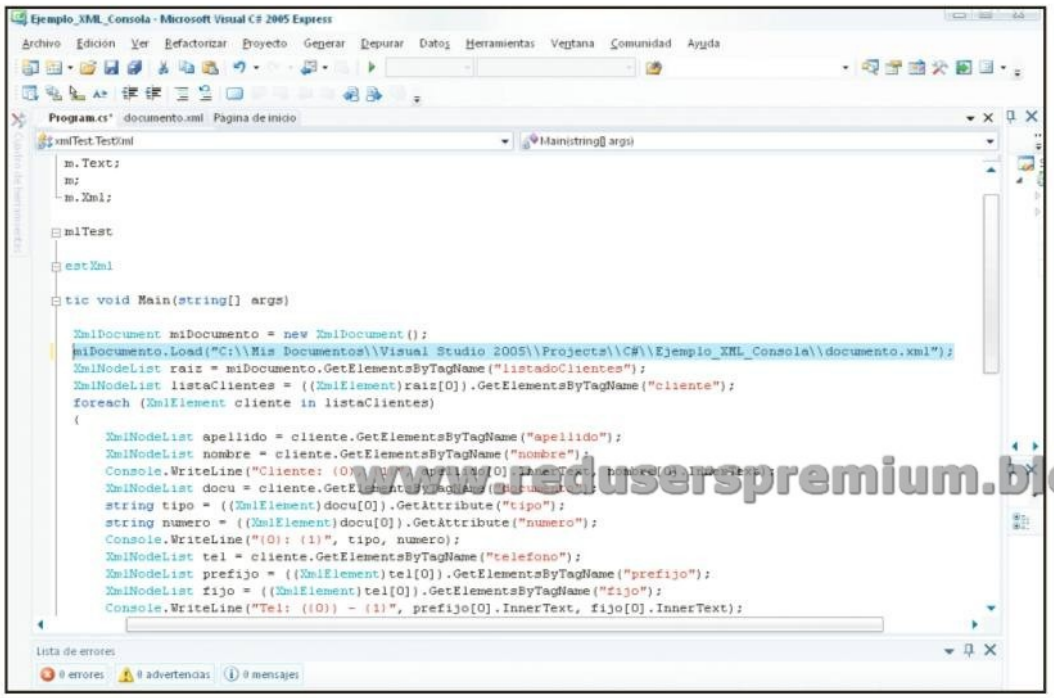


FIGURA 006 | Es bueno siempre darle la ruta completa al programa del archivo XML.

recurrimos al siguiente código:

```
XmlNodeList docu = cliente.GetElements
ByTagName("documento");
string tipo = ((XmlElement)docu[0]).
GetAttribute("tipo");
string numero = ((XmlElement)docu[0]).
GetAttribute("numero");
```

Y, por supuesto, lo enviamos a la consola de la siguiente manera:

```
Console.WriteLine("{0}: {1}", tipo, numero);
```

Nos resta por ver el tema del elemento telefono, que no es más que recorrer un nodo dentro del nodo en el que estamos, y esto puede ocurrir en **n** niveles. Lo importante es conocer siempre en dónde nos encontramos, ya que en algunos casos, suele ser bastante complejo debido a la longitud del documento, la cantidad de nodos, su profundidad y la cantidad de atributos de cada uno. En el caso que nos incumbe, declaramos un objeto para contener al nodo que alberga los diferentes datos del teléfono y, luego, uno para cada dato en particular.
Lo enviamos a la consola de este modo:

```
XmlNodeList tel = cliente.GetElements
ByTagName("telefono");
XmlNodeList prefijo = ((XmlElement)tel[0]).
GetElementsByTagName("prefijo");
XmlNodeList fijo = ((XmlElement)tel[0]).
GetElementsByTagName("fijo");
Console.WriteLine("Tel: ({0}) - {1}",
prefijo[0].InnerText, fijo[0].InnerText);
```

Hasta aquí hemos visto cómo se conforma un documento XML y de qué manera leerlo desde un lenguaje como C#. De forma parecida se logra crear un documento XML, agregarle nodos, atributos, etc. Sin adentrarnos en esta tarea, basta especificar que nuestro objeto XmlDocument expone métodos como CreateElement para agregar un elemento, y nuestros nodos, métodos como AppendChild para agregarle un elemento hijo (una nueva rama a la rama del árbol) y, así, ir creando un documento XML.

Esquemas y validaciones

Hemos visto que un documento XML es un medio estructurado para almacenar y compartir información. Como explicamos en el capítulo sobre bases de datos, un esquema describe la estructura en la que se alojarán los datos. En este punto debemos hacer una distinción muy importante entre un documento XML bien construido y uno válido, pues un documento puede estar correctamente construido en cuanto a la sintaxis XML, pero no corresponderse con el esquema que lo rige. Un esquema de datos establece restricciones en la

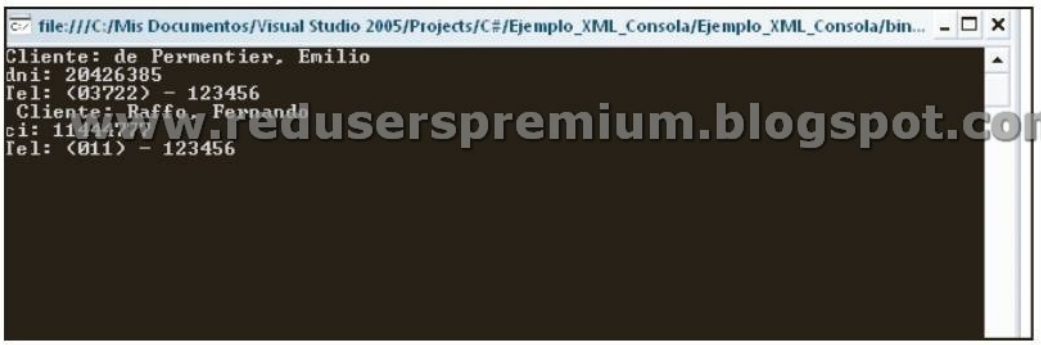


FIGURA 007 | Aquí tenemos el resultado de los registros cargados en el documento XML.

estructura del contenido del documento de dos maneras:

- Estableciendo el modelo de contenido de los documentos, lo que define el orden y el anidamiento de los elementos.
- Estableciendo el tipo de datos permitidos en cada caso dentro del documento.

Para esto utilizaremos los DTD, que pueden ser externos o internos. Pero antes de adentrarnos en este tema, hay que saber que no siempre son los más utilizados por la comunidad consumidora de XML, ya que la mayoría prefiere usar XML Schema.

Veamos un ejemplo de DTD:

```
<!ELEMENT clientes (cliente)+>
<!ELEMENT cliente (nombre, direccion+,
ciudad, provincia, telefono)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT telefono (fijo, movil?)>
<!ELEMENT fijo (#PCDATA)>
<!ELEMENT celular (#PCDATA)>
```

Esta tabla también se aplica a "group":

```
<?xml version="1.0" ?>
<!DOCTYPE clientes SYSTEM "clientes.dtd">
<clientes>
  <cliente>
     <nombre>de Permentier</nombre>
     <direccion>San Martín 4545</direccion>
     <ciudad>Posadas</ciudad>
     <provincia>Misiones</provincia>
     <telefono>
        <fijo>555-1212</fijo>
        <movil>15-555-1213</movil>
     </telefono>
  </cliente>
  <cliente>
     <nombre>Pérez</nombre>
     <direccion>Belgrano 4545</direccion>
     <ciudad>Mar del Plata</ciudad>
     <provincia>Buenos Aires</provincia>
     <telefono>
        <fijo>333-1212</fijo>
        <movil>15-333-1213</movil>
     </telefono>
  </cliente>
</clientes>
```

Observemos que lo primero que se hace en el documento, junto con la declaración de la versión de XML utilizada, es el enlace al DTD empleado. Este DTD brinda la posibilidad de conocer la mecánica básica acerca

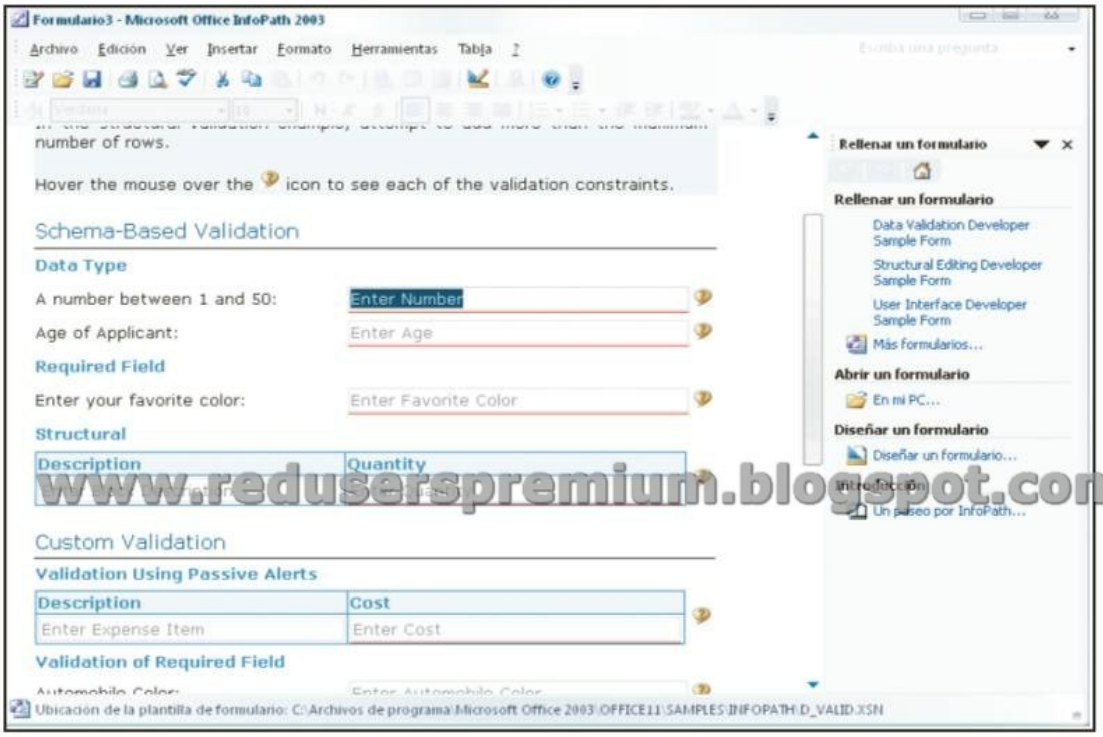
FIGURA 008 | Modelado del archivo clientes.dtd desde el Bloc de notas.

de cómo se define un modelo de documento. Lo primero que vamos a ver es el significado del signo más (+), que indica que la aparición del elemento afectado puede ocurrir más de una vez. Según esto, cuando definimos que existe un elemento raíz clientes, le indicamos que va a tener uno o varios elementos cliente. A su vez, cuando definimos los elementos que puede contener cliente, colocamos un signo + en dirección, lo que indica que puede poseer más de una dirección. Así definimos todos los elementos de nuestro documento y la anidación entre ellos, pero vamos a darle una mirada a telefono, en el que se indica que puede poseer dos elementos denominados "fijo" y "movil". En este caso vemos que luego de movil, aparece un signo de cierre de interrogación para indicar que el elemento es opcional, es decir que puede no aparecer dentro de nuestro documento.
Vemos que los elementos se enumeran señalando que contienen #PCDATA, lo cual significa datos de caracteres que serán analizados sintácticamente. En el caso del elemento telefono, no contendrá datos de caracteres, sino sólo los elementos permitidos, denominados en nuestro ejemplo como fijo y movil.

> Un documento XML puede estar correctamente construido en cuanto a sintaxis, pero no corresponderse con el esquema que lo rige.

A continuación, entraremos en el mismo modelado de datos del documento ejemplificado, pero con XML Schema. Este tipo de documentos utiliza un vocabulario llamado XML-Data. Y es aquí donde nos adentraremos más profundamente, ya que esta clase de validaciones van más allá de los DTD. Esto se debe a que XML Schema, por ejemplo, permite asociar tipos de datos a los elementos, pero lo más importante y curioso es que

FIGURA 009 | InfoPath de Microsoft permite crear formularios para almacenar datos en archivos XML.



Los DTD no siempre son los más utilizados por la comunidad de XML, ya que la mayoría prefiere usar XML Schema.

este tipo de documentos se escribe en formato XML.
Veamos un ejemplo:

```
<?xml version="1.0"?>
<Schema name="clientes"
 xmlns="urn:schemas-microsoft-com:xml-data"
 xmlns:dt="urn:schemas-microsoft-com:
 datatypes">
 <ElementType name="nombre" content=
 "textOnly"/>
 <ElementType name="direccion" content=
 "textOnly"/>
 <ElementType name="ciudad" content=
 "textOnly"/>
 <ElementType name="provincia" content=
 "textOnly"/>
 <ElementType name="fijo" content="textOnly"
 dt:type="int"/>
 <ElementType name="celular" content="textOnly"
 dt:type="int"/>
 <ElementType name="telefono" content=
 "eltOnly">
  <element type="fijo" minOccurs="1"
  maxOccurs="1"/>
  <element type="celular" minOccurs="0"
  maxOccurs="1"/>
 </ElementType>
 <ElementType name="cliente" content="eltOnly">
  <element type="nombre" minOccurs="1"
  maxOccurs="1"/>
  <element type="direccion" minOccurs="1"
  maxOccurs="2"/>
  <element type="ciudad" minOccurs="1"
  maxOccurs="1"/>
  <element type="provincia" minOccurs="1"
  maxOccurs="1"/>
  <element type="telefono" minOccurs="1"
  maxOccurs="1"/>
 </ElementType>
 <ElementType name="clientes"
 content="eltOnly">
  <element type="cliente" minOccurs="1"/>
 </ElementType>
</Schema>
```

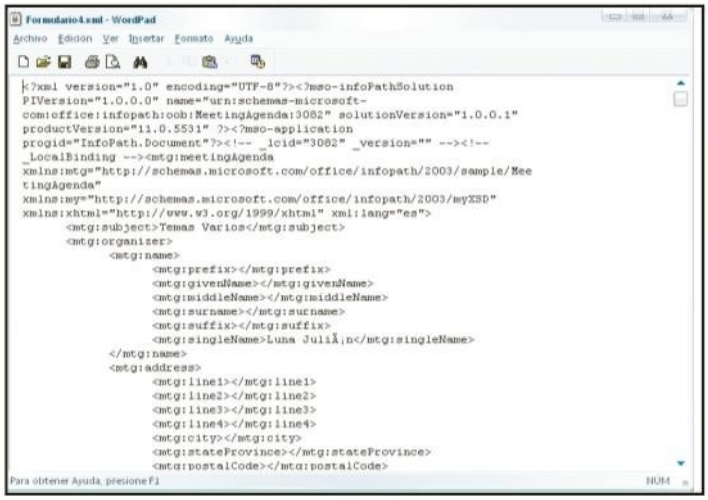


FIGURA 010 | La estructura de un archivo XML creado en InfoPath es similar a la de cualquier otro archivo XML.

XML Schema

XML Schema permite asociar tipos de datos a los elementos, escribiendo los documentos en formato XML.

Al analizar este archivo, vemos que no es tan compacto como un DTD, pero sí resulta bastante simple de entender.

Pasemos a ver en detalle qué significa cada parte, empezando por saber que primero se enumeran los elementos, y luego, los elementos que los incluyen. Notemos que definimos los elementos fijo y celular y, después, telefono,

que incluye a los dos anteriores; así vamos respetando la anidación desde adentro hacia la raíz.

A los primeros elementos, como nombre o direccion, les indicamos que contendrán texto. Esto no quiere decir que el tipo de dato sea un "string", sino que incluirán textos no analizables por el analizador de sintaxis de XML. Claro que si no especificamos el tipo de texto, podrá contener cualquier cadena de texto. Así es que podemos ver que los elementos fijo y movil tienen el agregado de definir el tipo de dato que incluirán.
Ahora nos centraremos en el elemento telefono, en el cual se indica que contendrá sólo elementos (eltOnly). Cabe aclarar que en nombre, por ejemplo, no puede contener elementos, ya que estaba declarado como textOnly. Al declarar un elemento con eltOnly, luego tenemos que especificar los elementos que incluirá, los cuales previamente debieron ser definidos. Aquí podemos observar otro elemento de definición, como minOccurs, que señala la cantidad mínima de veces en que ese elemento debe aparecer dentro de telefono, para este caso. De manera análoga, se define maxOccurs. En nuestro caso, vemos que "fijo" debe aparecer una y sólo una vez; por su parte, "celular" tiene que aparecer, como máximo, una vez, pero podría no aparecer, lo que es lo mismo que decir que no es de carácter obligatorio.
Veamos ahora algunas diferencias que existen entre el modelado de datos en DTD y XML Schema:

Los DTD poseen una sintaxis especializada, mientras que los esquemas XML están basados en la propia sintaxis de XML.

- Los DTD poseen una sintaxis especializada, mientras que los esquemas XML están basados en la propia sintaxis de XML.
- Los esquemas XML están basados en XML, por lo que se pueden validar como documentos XML.
- Los esquemas XML soportan varios tipos de datos, como int, float, boolean y date, entre otros.
- En los esquemas XML se presenta un modelo de datos abierto, lo cual permite ampliar su propio vocabulario y establecer una suerte de herencia entre los distintos elementos.

Tabla 3 | Composición del vocabulario de XML Schema

Elemento	Descripción
Schema	Sirve como elemento raíz del documento XML Schema.
Datatype	Describe los tipos de datos de elementos y atributos.
ElementType	Describe un tipo de elemento.
Element	Identifica un elemento que puede aparecer en otro tipo de elemento.
Group	Organiza los elementos en grupos con un mero fin organizativo.
AttributeType	Describe un tipo de atributo.
Attribute	Identifica un atributo que puede aparecer dentro de un elemento.
Description	Proporciona una descripción o documentación acerca de un elemento o atributo.

En la Tabla 3 vemos los diferentes elementos que componen el vocabulario de XML Schema y su descripción.

El elemento Schema siempre actúa como contenedor del resto del documento XML Schema, e incluye dos atributos que utiliza para establecer el nombre del esquema, que es name, y el espacio de nombres correspondiente, que es xmlns.
En cuanto a dataType, es el que se encarga de definir el verdadero contenido de los elementos y atributos. Posee el atributo dt:type que lo describe, y sus valores posibles son: entity, entities, enumeration, id, idref, idrefs, nmtoken, nmtokens, notation, string, bin.base64, bin.hex, boolean, char, date, dateTime, dateTime.tz, fixed.14.4, float, int, number, time, time.tz, i1, i2, i4, r4, r8, ui1, ui2, ui4, uri y uuid. En nuestro documento podemos ver el tipo de dato "int".

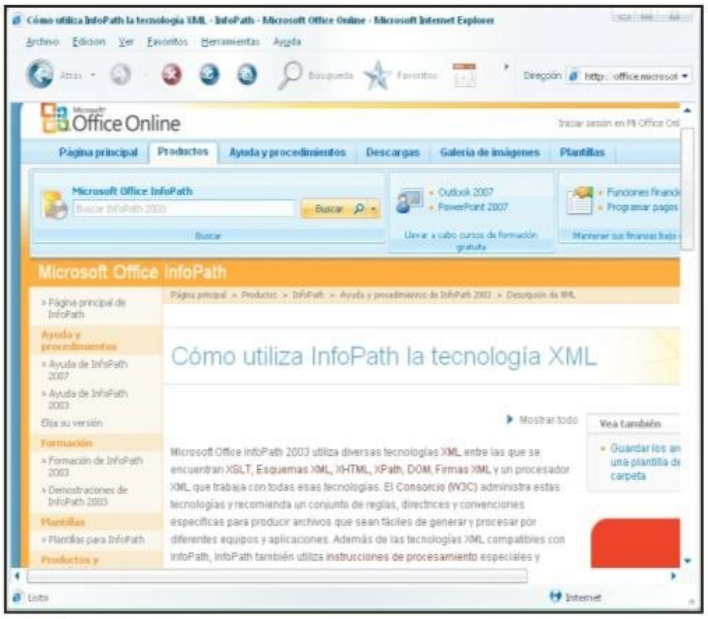


FIGURA 011 | La Web de Microsoft que cuenta sobre InfoPath, producto disponible en la suite Office.

Cuando hablamos de ElementType, vemos que posee un atributo name, en el cual se indica el nombre que identifica al elemento, y content, el contenido correspondiente. Para content podemos observar las siguientes posibilidades: empty, textOnly, eltOnly y mixed.
En el caso de element encontramos type, que indica el tipo de elemento que se está describiendo, y las posibilidades minOccurs y maxOccurs, que hemos tratado con anterioridad. En la Tabla 4 se presentan las diferentes relaciones entre estos dos últimos atributos y lo que significa realmente.

De la misma manera en que se definen y establecen los elementos, se trabaja con los atributos. Éstos, a su vez, pueden contener un tipo "default", que indica el valor por defecto del atributo, y "required", que indica si éste es de carácter obligatorio.

Tabla 4 | Relaciones entre atributos

minOccurs	maxOccurs	Número de veces en que el elemento puede producirse
0	1	0 ó 1
1		
0	*	Cualquier número de veces
1	*	Al menos una vez
>0	*	Al menos minOccurs veces
> maxOccurs	>0	0
Cualquier valor	< minOccurs	0

Argentina $8,90 (recargo al interior $0,20) / México: $45

USERS

Microsoft

Curso teórico y práctico de programación

Desarrollador .net

Con toda la potencia de **Visual Basic .NET** y **C#**

La mejor forma de aprender a programar desde cero

★★★★★

Basado en el programa Desarrollador Cinco Estrellas de Microsoft

14

XML
Fundamentos - Sintaxis
Esquemas y validaciones - XPath

SALIDA LABORAL
La mejor preparación para rendir los exámenes
GARANTIZADA

ADO.NET avanzado
DataSet - System.Data
La cadena de conexión

ISBN 978-987-1347-43-8
00014
9 789871 347438