UNIDAD DIDÁCTICA 3

MANIPULACIÓN DE DOCUMENTOS WEB

MÓDULO PROFESIONAL:
LENGUAJE DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN

Índice

RESUMEN INTRODUCTORIO

En esta unidad estudiaremos la manipulación de documentos web, haciendo un recorrido por los distintos lenguajes de scripts de cliente, su estructura, acceso y tratamiento de los distintos elementos, centrándonos en el caso del metalenguaje XML, como ejemplo concreto, dada la gran aceptación y uso en el desarrollo web.

INTRODUCCIÓN

Cuando hablamos de desarrollo web en el sentido de creación de documentos con el código fuente que permita publicar una página o sitio web, resulta evidente la necesidad de contar con un lenguaje de programación que nos permita realizar ese desarrollo web de forma efectiva.

Dentro de los lenguajes de programación, encontramos los lenguajes de script del lado cliente, que permiten una manipulación de los contenidos por parte de los desarrolladores, existiendo en el mercado gran cantidad de lenguajes como son Java, Phyton o JavaScript, entre otros.

La manipulación y acceso a sus elementos para realizar acciones como creación o eliminación de estos, será imprescindible a la hora de crear páginas o sitios web, encontrando en XML, un ejemplo con el que poder ilustrar esa manipulación de documentos web.

CASO INTRODUCTORIO

Trabajas en una empresa que se encarga del mantenimiento del sitio web, para lo que debes contar con conocimientos de programación y de los lenguajes de script de cliente, para su utilización en el desarrollo web.

Se ha planteado la necesidad de reordenar el sitio web corporativo con nuevas funcionalidades mediante la modificación de los elementos que ya existen en XML, así como otros tipos de acciones relacionadas que implicarán la creación o eliminación de algunos de ellos y sus atributos.

Al finalizar el estudio de la unidad, conocerás el concepto de lenguaje de script de cliente, serás capaz de reconocer los principales lenguajes de script de cliente y de realizar un documento XML.

1. LENGUAJES DE SCRIPT DE CLIENTE. CARACTERÍSTICAS Y SINTAXIS BÁSICA. ESTÁNDARES

En tu labor dentro del departamento de informática de la empresa, actualmente estás directamente implicado en el desarrollo de las nuevas páginas web del sitio corporativo. Por lo que, será necesario que conozcas qué es un lenguaje script de cliente, algunos de los más utilizados y saber sus características para elegir el más adecuado, además de conocer la sintaxis básica y los estándares del lenguaje para un desarrollo correcto.

Los lenguajes de programación del lado del cliente, se utilizan en la creación de páginas web, de modo que sus instrucciones, sentencias o código, va insertado dentro de un documento HTML, que posteriormente se ejecuta sin necesidad de compilación. Algunos de los lenguajes de programación script más conocidos son Javascript, HTML, CSS o VB script.

Entre sus características podemos destacar:

- Son lenguajes que no necesitan ser compilados, sólo interpretados para la ejecución de su código.
- No es necesario declarar variables para la asignación de tipos de datos, son deducidos dentro del contexto que se utilizan, asignándole el valor en ese momento.
- La ejecución de acciones se realizará de acuerdo con las instrucciones que se especifiquen en el código fuente.
- Se busca la simplicidad y eficacia mediante la sencillez del código fuente en los scripts.
- Posibilidad de utilizar scripts para automatización de tareas, mantenimiento de servidores, etc.
- Liberan al servidor de carga de procesamiento.
- Disminuye el tráfico de red.
- Permite la creación de páginas web dinámicas en el cliente y validación de formularios.

En general, la programación del lado cliente nos permite la validación de algunos datos, antes de que sean enviados al servidor web, pudiendo obtener informes de forma inmediata sin necesidad de envío de un formulario, por ejemplo.

Destacamos una serie de ventajas de este tipo de programación, como son:

- Facilidad de depuración de código sin necesidad de nueva compilación.
- Ejecución de script por el intérprete.
- Fácil mantenimiento y modificación del código.

Sintaxis básica

Los lenguajes de programación del tipo script cuentan con una serie de reglas que deben ser tenidas en cuenta para la escritura del código fuente para una sintaxis correcta.

Todos los lenguajes tienen una sintaxis muy similar, nos centraremos en el caso concreto de JavaScript, donde citaremos las principales normas básicas de sintaxis:

1. Los espacios y líneas en blanco no son tenidos en cuenta, lo que permite una escritura de código ordenada y clasificada según el desarrollador, para una mejor comprensión.
2. Se distinguen entre el uso de mayúsculas y minúsculas, dato a tener en cuenta porque el script daría error.
3. No es necesario la definición del tipo de dato cuando se crean variables.
4. La terminación de una sentencia no necesita escribir punto y coma (;), aunque resulta recomendable por sí ser necesario en otros lenguajes de programación y facilita su lectura.
5. Admite la incorporación de comentarios dentro del código fuente, tanto en una única línea como en varias líneas.

Estándares

Cuando tratamos de que el proceso de navegación web tenga un correcto funcionamiento y operatividad, es necesario que los lenguajes de programación utilizados cumplan con una serie de estándares, que quedan recogidos en ECMAScript.

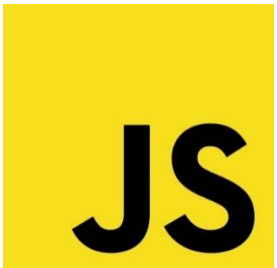
ECMAScript, es el estándar que desde el año 2015 rige cómo debe ser el funcionamiento de JavaScript y cómo debe ser interpretado, con incorporaciones que podemos ver en la siguiente tabla.

Año	Novedad incorporada
2015	ECMAScript 6, mejora en sintaxis y estructura de iteración
2016	ECMAScript 7, mejoras en el operador de exponenciación y matrices
2017	ECMAScript 8, incluye constructores async/await, que funcionan usando generadores
2018	ECMAScript 9, se agregan los operadores rest/spread para variables
2019	Nuevas características a través de funciones como Array.flat(), String.trimStart() o la posibilidad de utilizar en el manejo de errores try / catch
2020	Incorpora nuevas funciones e incluye el tipo primitivo BitInt
2021	Operadores de asignación lógica, separadores numéricos y referencias débiles.
2022	Variaciones en el uso del operador "await", nuevas declaraciones de instancias y cambios en clases y métodos.
2023	Cambio de matriz por copia, cadenas Unicode bien formadas y gramática hashbang

Entre la amplia oferta de lenguajes existentes, a continuación, citamos algunos de los más representativos, sobre todo por su gran comunidad de usuarios y popularidad.

1. JavaScript

Es quizá, el lenguaje de script de cliente, es decir, se ejecuta en el navegador web del usuario final, más popular del mundo web. Entre sus muchas funcionalidades podemos destacar la posibilidad de interactuar con los elementos de una página web, a través de la manipulación del Document Object Model (DOM), pudiendo también llevar a cabo el manejo de eventos, comunicaciones con los servidores, creación de animaciones, formularios, etc.



Logo JavaScript.
Fuente: https://es.logodownload.org/javascript-logo/

Se caracteriza por su flexibilidad y dinamismo, que favorece la creación de sitios interactivos con contenidos de calidad por parte de los desarrolladores.
Utilizado en distintos tipos de software que afecta a distintas áreas como pueden ser aplicaciones web, juegos, software específico, o blockchain, y por supuesto en el desarrollo web.

Entre sus características destacan:

- Su ejecución se realiza de manera principal en navegadores web.
- Se interpreta (no compila) línea a línea.
- Flexibilidad y facilidad de desarrollo.
- Posibilidad de ejecución en distintas plataforma y sistemas operativos.
- No necesita instalación para actualizaciones o distribución, basta con cargar la nueva versión en los navegadores web.
- Permite la implementación de conceptos orientados a objetos.
- A través de DOM, es posible trabajar con elementos CSS y HTML.
- Su ejecución es una respuesta a un evento del sistema o una acción del usuario.
- Es asíncrono respecto a la realización de tareas sin el bloqueo del código.

NOTA DE INTERÉS

JavaScript no es lo mismo que Java, son lenguajes de programación diferentes con finalidades distintas.

2. Python

Creado en 1991 por Guido van Rossum, Python, es un lenguaje de script de cliente muy potente al disponer de estructuras de datos de alto nivel eficientes, y un sistema de programación orientado a objetos, que lo convierten en un lenguaje ideal para la programación mediante scripts (scripting) y el desarrollo web en cualquier áreas y multitud de plataformas.

Logo Python.
Fuente: https://rootstack.com/es/technology/python-tecnologia

Entre sus características destacan:

- Es un lenguaje de propósito general, pensado principalmente para la creación de páginas web, pero no es un fin exclusivo.
- De libre distribución.
- Amplia biblioteca de librerías y funciones ya predeterminadas.
- Gran comunidad para soporte, información y consulta.
- Permite no sólo la programación orientada a objetos, sino también otros tipos de programación como a través de sentencias de bucle (imperativa) o con módulos y funciones (funcional).
- Es un tipo de lenguaje interpretado sin necesidad de compilación.
- Multiplataforma: Windows, Unix, Linux, Mac Os, etc.
- Es de tipado dinámico, en la declaración de variables, no es necesario que se especifique el tipo de dato, éste será recogido del propio programa a la hora de su ejecución.
- Orientado a objetos.

ENLACE DE INTERÉS

El intérprete de Python está disponible en su página web oficial.

3. PHP

Creado en 1995 por Rasmus Lerdorf, este tipo de **lenguaje de script de servidor** fue diseñado originariamente para el desarrollo de sitios web dinámicos, pudiendo ser incluido directamente en el código HTML sin necesidad de utilizar un archivo externo.

Logo PHP.
Fuente: https://nestrategia.com/que-es-php/

Destacan como características, las siguientes:

- Software libre.
- Sencilla instalación.
- Fácil curva de aprendizaje.
- Trabaja con bases de datos de forma dinámica, como son MySQL, Oracle, SQL Server, entre otras.
- Programación confiable y segura.
- Gran comunidad para consulta y documentación.
- Ampliación mediante módulos o extensiones.
- Manejo de excepciones.
- Utilización de variables sin definir tipo.
- Es multiplataforma.

Disponible su descarga desde la página https://www.php.net/

4. XML

XML (acrónimo en inglés de e**X**tensible **M**arkup **L**anguage, "lenguaje de marcas extensible") es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas o aplicaciones. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. Se trata de una tecnología sencilla, que permite la compatibilidad entre sistemas para compartir información de manera segura, fiable y fácil.

Al igual que el HTML, se basa en documentos de texto plano en los que se utilizan etiquetas para delimitar los elementos de un documento.

Con XML se buscará representar la información estructurada, es decir, una información que se compone de partes bien definidas que, a su vez, se componen de otras partes. Por tanto, es posible formar un árbol de partes de información.

XML proviene de un lenguaje inventado por IBM en los años 70, llamado GML (General Markup Language), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información.

A diferencia de un documento HTML, que contiene datos mal definidos, es decir, mezclados con elementos de formato, un documento XML contiene solamente datos que se autodefinen. En XML el contenido queda separado totalmente de la presentación.

Los documentos XML siguen una estructura estrictamente jerárquica en lo que respeta a las etiquetas que delimitan sus elementos. Por tanto, las etiquetas deberán estar incluidas en el ámbito de otras. Además, los elementos con contenido deberán cerrarse correctamente.

Un aspecto importante es que los documentos XML sólo permiten un elemento raíz, del que derivarán todos los demás. Es decir, en un documento XML bien formado, la jerarquía de los elementos sólo tendrá un elemento inicial.

Por ejemplo, en este caso, <productos> sería el elemento inicial:

```
<?xml version=”1.0” encoding=”uft-8”?>
<productos>
<articulo>
<nombre>Refresco 1LT</nombre>
<precio>1.25</precio>
</articulo>
```

```
<articulo>
<nombre>Agua 1LT</nombre>
<precio>0.85</precio>
</articulo>
</productos>
```

EJEMPLO PRÁCTICO

Luis, jefe del departamento de informática de su empresa, es el encargado de elegir el tipo de software tanto a nivel aplicaciones, sistemas operativos, diseño, etc., por lo que en el caso de desarrollo web debe realizar una propuesta.

Dentro de la amplia oferta de lenguajes de programación tipo script de lado cliente que existen en la actualidad, debe realizar una recopilación de los más extendidos por uso y comunidad de usuarios con el fin de poder elegir cuál de ellos se utilizará a partir de ahora en los nuevos desarrollos web.

¿Qué lenguajes serían los que podría proponer?

Solución.

Luis, en la actualidad, podría proponer entre los siguientes:

JavaScript. Lenguaje de programación interpretado, no necesita compilación, que permite la implementación de funciones complejas en páginas web con actualizaciones dinámicas.

Python. Lenguaje de programación de alto nivel utilizado en el desarrollo de todo tipo de aplicaciones, incluidas las aplicaciones web. Es de tipo interpretado, sencillo y multiplataforma.

PHP. Lenguaje de programación interpretado de código abierto que permite ser incrustado en HTML, muy utilizado para el desarrollo de páginas y sitios web.

XML. Es un lenguaje de marcado que define un conjunto de reglas de codificación de documentos web que permiten su lectura tanto a nivel humano como por máquina. Establece estructura, formato y normas a seguir por otros lenguajes.

VÍDEO DE INTERÉS

Este vídeo nos presenta una comparativa entre lenguaje compilado (C#), lenguaje interpretado (Python) y Java.

2. SELECCIÓN Y ACCESO A ELEMENTOS

En esa labor de desarrollo y programación de las páginas web, te han pedido que profundices un poco más sobre las recomendaciones generales de sintaxis para los lenguajes de programación, sobre todo, a la hora de saber cómo gestionar los elementos que vas a incorporar, a nivel selección y acceso, así como los posibles atributos que puedes utilizar.

En el tratamiento de elementos dentro de un lenguaje de programación, suele ocurrir igual que en su sintaxis, es bastante común la forma de trabajar con ellos, de este modo, vamos a seguir el ejemplo en XML, como metalenguaje más utilizado.

Comenzaremos por lo que constituye un documento XML, de tal modo que, la parte inicial de un documento XML, se puede usar para especificar una serie de propiedades que tendrá dicho documento. Para ello, se utilizan unas marcas específicas, diferentes de las usadas en el resto del archivo para especificar los datos. Todas estas marcas deben comenzar con: "<?" y terminan "?>". Se trata pues, de instrucciones que se utilizan para indicar que debe hacerse un determinado procesamiento a los datos. Es importante escribir estas etiquetas para el uso aplicaciones específicas para el tratamiento de los datos.

El primer elemento que debe tener un archivo XML es lo que se denomina declaración. Se trata de una zona en la que se indicarán ciertas definiciones que afectarán al documento. Esta declaración debe ser la primera línea del archivo XML, no pudiendo estar precedida tampoco por espacios o saltos de línea.

La declaración debe aparecer encerrada entre "<?xml" y "?>". De forma obligatoria, es necesario que aparezca la versión del estándar XML que se usará en dicho documento. Hoy día es la 1.0. Por tanto, quedará:

```
<?xml version="1.0"?>
```

Además, es posible añadir a la declaración otra información, como la codificación usada en el documento y si se utiliza un archivo para validar el documento. Los atributos utilizados serían los siguientes:

- **encoding:** opcional, por defecto se usa UTF-8, pero también es posible usar codificaciones regionales. En el caso de España, se podría usar también la codificación ISO-8859-1 o la ISO-8859-15, que cuentan con los caracteres latinos como vocales con tildes, letra ñ, signos de apertura de exclamación e interrogación, etc.

- **standalone:** también de carácter opcional, se usa para indicar si el documento hará referencias a otros documentos externos o no. Si no lo hace, es decir, el documento se deberá considerar de forma aislada, por lo que se asignará a este atributo el valor "yes".

Así pues, un ejemplo más completo con todos estos datos podría ser:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Seguidamente a la declaración del archivo XML, se pueden realizar otras como, por ejemplo, la de los ficheros de estilo que se usarán en la representación del documento:

```
<?xml-stylesheet ref="simple-ie5.xsl" type="text/xsl" ?>
```

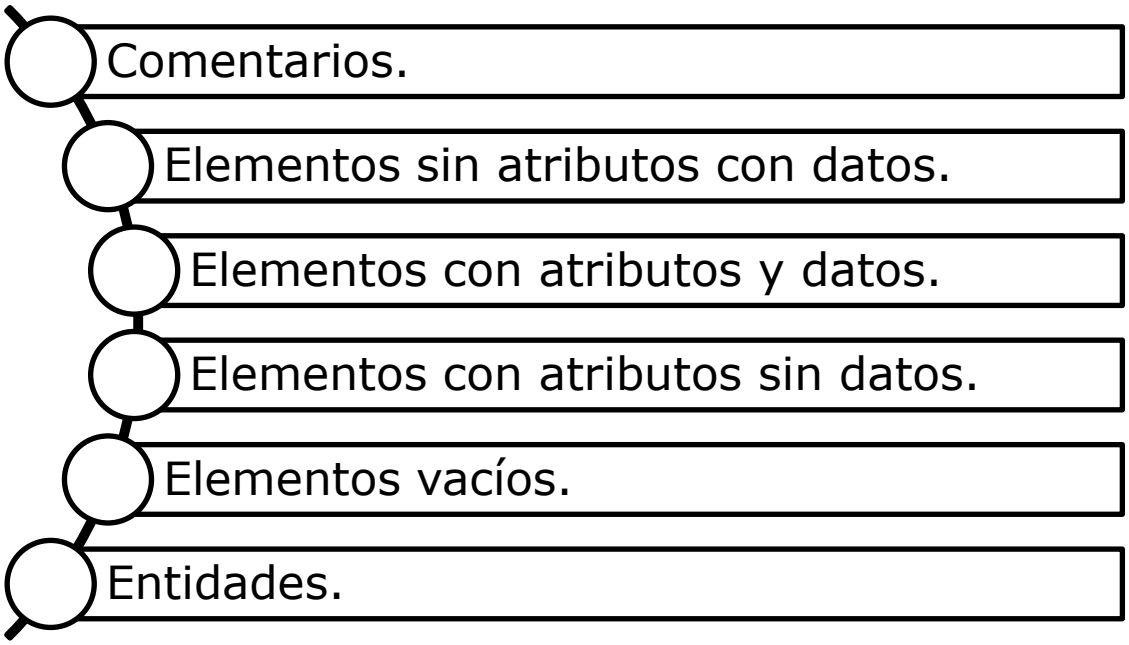
Más adelante, se verá cómo usar dichos ficheros.

ENLACE DE INTERÉS

En esta página puedes encontrar la referencia oficial del estándar XML:

2.1 Elementos de un XML

Un documento XML puede contener los siguientes elementos:



Comentarios

Como en todo lenguaje, en XML se pueden insertar comentarios. El comentario tiene la misma estructura que en HTML, comenzando por los caracteres "<!--" y terminando por "-->".

Ejemplo:

```
<ejemplo>
<datos>En este ejemplo se incluye un comentario</datos>
<!—este es el comentario incluido -->
</ejemplo>
```

Elementos sin atributos y con datos

Son los elementos compuestos únicamente por datos encerrados entre etiquetas. Las etiquetas tienen el mismo formato que en HTML, es decir, un nombre entre signos de < y >.

A diferencia de las etiquetas HTML, XML no define unas determinadas etiquetas, sino que será la persona encargada de especificar el formato de cada XML en particular, quien otorgue un nombre a cada una de esas etiquetas en función del contenido que vaya a tener en su interior. Normalmente, se suele dar un nombre descriptivo de dicho contenido. Por ejemplo, si se quiere almacenar una matrícula de un coche, un buen nombre para la etiqueta podría ser: <matricula>. Y un elemento formado por dicha etiqueta podría ser:

```
<matricula>0000-AAA</matricula>
```

Elementos con atributos y datos

Además de las etiquetas de apertura y cierre, los elementos también pueden contener información dentro de las propias etiquetas, entre sus caracteres de apertura y cierre. Se definen de igual forma que los atributos que se estudiaron para las etiquetas HTML, es decir, de la forma: atributo=valor. No obstante, en este caso, el valor del atributo debe estar obligatoriamente entre comillas.

Al igual que se ha comentado para las etiquetas, los nombres de los atributos no vienen dados por ninguna especificación del estándar, sino que serán establecidos en el momento de realizar la especificación del formato de dicho archivo, usando también, preferiblemente, nombres descriptivos del valor que podrán tener asociado.

Por ejemplo, para almacenar notas de un empleado a otro, se podría definir un esquema como el siguiente:

```
<nota de="Juan" para="Jose" prioridad="alta">
    <mensaje>Llamar a Luis</mensaje>
    <tlf>
        <casa>9520000000</casa>
        <movil>600000000</movil>
</tlf>
</nota>
```

Elementos con atributos sin datos

En el punto anterior se han introducido los elementos con atributos y se ha visto que los atributos son por si solos capaces de ofrecer información. Por lo tanto, no es imprescindible usar datos para almacenar información, sino que puede hacerse usando atributos.

Por ejemplo, otra especificación diferente del formato del archivo anterior también serviría para almacenar la misma información:

```
<nota de="Juan" para="Jose" prioridad="alta" mensaje="Llamar a Luis"
tlfcasa="952000000" tlfmovil="652360578"></nota>
```

Y, siguiendo con el primer ejemplo, también sería posible almacenar un número de matrícula. En este caso, dentro de una etiqueta que se podría llamar coche:

```
<coche matricula="0000-AAA"></coche>
```

Elementos vacíos

En algunas ocasiones puede que no interese indicar ningún valor para alguno de los elementos. Por ejemplo, en el caso de las notas, puede ser que la persona a la que hay que llamar, no haya dejado un número de fijo o de móvil. En ese caso, se dejaría el elemento vacío.

Un elemento vacío se puede representar de dos formas. Como se ha visto hasta ahora, es decir, escribiendo las etiquetas de apertura y cierre sin incluir en medio ningún contenido: <elemento></elemento>, o bien de modo abreviado: <elemento/>

Entidades

Las entidades son unidades de almacenamiento, y pueden contener desde un carácter hasta el nombre de un fichero que se quiera incluir en el XML. De forma similar a las entidades predefinidas que se utilizaban en HTML para mostrar ciertos caracteres problemáticos, las entidades se escriben de la forma: &entidad;

A diferencia de las entidades vistas para HTML, éstas no tendrán un nombre y un valor determinados especificados por el estándar, sino que se definirán según interese. Es decir, se comportan como las constantes en programación, y se usan para evitar escribir repetidamente lo mismo.

Por ejemplo, si se tratara de almacenar datos de un contrato y en varios puntos del documento se debe especificar la fecha del día de su formalización, se puede escribir inicialmente una fecha cualquiera y reemplazarla al finalizar de editar el documento, o se puede utilizar una entidad, y así sólo se deberá modificar el valor de la entidad al finalizar el documento. Son como la representación de constantes de otros lenguajes, pero para usarlas dentro de un documento XML.

Otro uso muy habitual de las entidades es la sustitución de cadenas de texto excesivamente largas para no repetirlas a lo largo del documento. Por ejemplo, a &soap; puede asignársele el valor “Simple Object Access Protocol”, y así cada vez que se necesitara escribir dicha cadena, bastaría con escribir únicamente &soap;.

Las entidades, también pueden usarse para representar caracteres extraños, como se hacía en HTML.

3. CREACIÓN Y MODIFICACIÓN DE ELEMENTOS

Sigues con tu labor de creación de páginas web. En este momento, ampliando información sobre los elementos que puedes incorporar en los documentos XML y otro factor importante es que conozcas qué acciones puedes realizar con ellos, como, por ejemplo, su creación y modificación. De este modo, podrás cubrir las necesidades demandadas para el nuevo desarrollo web.

Continuando con el caso de XML, la creación de elementos debemos crear un diccionario propio para definirlo, lo cual se conseguirá mediante un fichero DTD (Document Type Definition).

Una DTD tiene como misión definir de manera estructurada todas las partes que van a constar en el documento XML, siendo necesario un procesador XML que permita validar el documento mediante la etiqueta DOCTYPE.

La DTD puede ser interna mediante la etiqueta anteriormente citada o externa donde irá referenciado el documento externo dentro de la etiqueta.

A continuación, ya será posible la declaración de los distintos elementos, con atributos o sin ellos, dentro del documento XML, mediante la declaración del tipo de elemento que comenzará con la etiqueta <!ELEMENT.

Un ejemplo de DTD interna sería:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores [
   <!ELEMENT marcadores (pagina)*>
   <!ELEMENT pagina (nombre, descripcion, url)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT descripcion (#PCDATA)>
   <!ELEMENT url (#PCDATA)>
]>

<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

Ejemplo DTD interna.

Fuente: https://www.abrirllave.com/dtd/documento-xml-asociado-a-una-dtd-interna.php

En el caso de esta DTD, conviene explicar un poco su contenido:

- Se establece “marcadores” como elemento raíz del documento XML, que puede contener un número de páginas que va de cero a varias, indicado en el código mediante (página)*.
- Definimos tres elementos hijos dentro de cada página, que serán “nombre”, “descripción” y “url”, mediante el código “página (nombre, descripción, url)”.
- Podemos indicar que los elementos, en este caso nombre, descripción y url, pueden contener texto, al escribir entre paréntesis #PCDATA, que puede ser analizado por el procesador de XML.

En el caso de una DTD externa, podemos encontrar dos tipos: privada y pública.

Comenzamos por ver la DTD privada donde deberemos utilizar SYSTEM en el elemento raíz:

```
<!DOCTYPE elemento-raíz SYSTEM "URI"
```

Partimos de un supuesto en el que definimos un archivo DTD con el nombre "identificadores.dtd" con el siguiente código en la DTD:

```
<!ELEMENT identificadores (pagina)*>
<!ELEMENT pagina (autor, titulo, edicion)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT edicion (#PCDATA)>
```

El documento XML resultante sería una lista de identificadores de libros con datos como el nombre, título y resumen, sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE identificadores SYSTEM "identificadores.dtd">

<identificadores>
   <pagina>
      <autor>Arturo Pérez Reverte</autor>
      <titulo>Capitán Alatriste</titulo>
      <edicion>2016/</edicion>
   </pagina>
   <pagina>
      <autor>Gabriel García Márquez</autor>
      <titulo>Cien años de soledad</titulo>
      <edicion>2017/</edicion>
   </pagina>
   <pagina>
      <autor>Antonio Muñoz Molina</autor>
      <titulo>No te veré morir</titulo>
      <edicion>2023/</edicion>
   </pagina>
</identificadores>
```

En el caso de la DTD pública se utilizará PUBLIC en el elemento raíz:

```
<!DOCTYPE elemento-raíz PUBLIC "identificador-público"
"URI"
```

Podemos utilizar un identificador público formal (FPI-Formal Public Identifier) que establece la W3C en sus estándares, que sería del tipo -//W3C//DTD XHTML 1.0 Strict//EN, quedando del siguiente modo:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE html PUBLIC>"-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1-strict.dtd>
<html>
   <head>
      <title>Título</title>
    </head>
   <body>
      <p>Resumen</p>
    </body>
</html>
```

¿SABÍAS QUE...?

Cuando se utiliza un archivo DTD, debe examinarse su contenido para ver qué atributos permite. Los nombres de atributo, como los nombres de etiqueta, deben ajustarse al archivo DTD.

EJEMPLO PRÁCTICO

Andrea, pertenece al grupo de desarrolladores del departamento de informática de su empresa, que se encarga de realizar proyectos web para clientes externos que solicitan sus servicios en la creación de páginas y sitios web corporativos.

En el proyecto que está inmersa ahora le han solicitado que realice un DTD de XML para la alimentación de datos de una pequeña biblioteca que van a incorporar en la empresa y poner a disposición de sus trabajadores.

En el archivo deben recogerse el título, autor y año de edición de los libros de la biblioteca.

¿Cómo sería el código fuente del DTD?

Solución.

Definimos un archivo DTD con el nombre “identificadores.dtd” con el siguiente código en la DTD:

```
<!ELEMENT identificadores (pagina)*>
<!ELEMENT pagina (autor, titulo, edicion)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT edicion (#PCDATA)>
```

VÍDEO DE INTERÉS

Conoce cómo crear una DTD a partir de un XML con Netbeans en este vídeo.

4. ELIMINACIÓN DE ELEMENTOS

Tu jefe te ha pedido, en esta ocasión, realizar cambios de interés o necesidades dentro del desarrollo web. Para ello, deberás conocer cómo poder modificar los elementos ya incluidos en el código fuente e incluso, si llega el caso, su eliminación. Circunstancia que se te irá planteando a lo largo de las diferentes revisiones y versiones de los documentos XML.

En el árbol XML puede realizarse la eliminación de elementos y atributos, así como otros tipos de nodos, de un modo sencillo. Simplemente teniendo la precaución de que la eliminar colecciones de elementos o atributos, debe materializarse con antelación una lista, para poder proceder posteriormente a su eliminación.

XML pone a nuestra disposición el método de extensión “Remove”, ubicado en el espacio de nombres dentro de “System.Xml.Linq” y ensamblado en “System.Xml.XDocument.dll”.

Su utilización permite:

1.- Quitar el atributo de su elemento primario, excepto si el elemento primario es null.

```
public void Remove ();
```

En una lista de cinco atributos, vamos a eliminar uno de ellos:

```
XElement root = new XElement("Root",
  new XAttribute("Atributo1", 1),
  new XAttribute("Atributot2", 2),
  new XAttribute("Atributo3", 3),
  new XAttribute("Atributo4", 4),
  new XAttribute("Atributo5", 5)
);
XAttribut att = root.Attribute (“Atributo3”);
att.Remove();
Console.WriteLine(root);
```

2.- Quitar todos los atributos de la colección de origen de su elemento primario, mediante la sentencia:

```
public static void Remove (this
System.Collections.Generic.IEnumerable<System.Xml.Linq.X
Attribute?> source);
```

Deberá realizarse la llamada a la colección de atributos creando una lista, y a continuación se pueden eliminar, la sentencia sería:

```
XElement root = new XElement("Root",
    new XAttribute("Atributo1", 1),
    new XAttribute("Atributo2", 2),
    new XAttribute("Atributo3", 3),
    new XAttribute("Atributo4", 4),
    new XAttribute("Atributo5", 5)
);
IEnumerable<XAttribute> atList =
    from at in root.Attributes()
    where (int)at >= 3
    select at;
atList.Remove();
Console.WriteLine(root);
```

3.- Quitar todos los nodos de la colección de origen de su nodo primario, mediante la sentencia:

```
public static void Remove<T> (this
System.Collections.Generic.IEnumerable<T?> source) where
T : System.Xml.Ling.XNode;
```

Deberá realizarse la llamada a la colección de atributos creando una lista, y a continuación se pueden eliminar, la sentencia sería:

```
XElement root = new XElement("Root",
    new XAttribute("Atributo1", 1),
    new XAttribute("Atributo2", 2),
    new XAttribute("Atributo3", 3),
    new XAttribute("Atributo4", 4),
    new XAttribute("Atributo5", 5)
);
IEnumerable<XElement> elList =
    from at in root.Elements()
    where (int)el >= 3
    select el;
elList.Remove();
Console.WriteLine(root);
```

4.- Quitar todos los atributos de un element concreto:

```
public void RemoveAll ();
```

Creamos un element con atributos y elementos secundarios, y, a través de la llamada la método, es posible eliminar los atributos y elementos secundarios:

```
XElement root = new XElement("Root",
    new XAttribute("Atributo1", 1),
    new XAttribute("Atributot2", 2),
    new XAttribute("Atributo3", 3),
    new XElement("Could1", 1),
    new XElement("Could2", 2),
    new XElement("Could3", 3),
);
Root.RemoveAll();
Console.WriteLine(root);
```

5.- Quitar o agregar un valor a un elemento secundario:

```
public void SetElementValue (System.Xml.Ling.Xname
name, object? value);
```

El parámetro name se pasará dentro de XName y será el que queremos cambiar, y el valor a asignar al elemento secundario se eliminará si es null, asignándose en caso contrario mediante una representación de cadena, y asignando el valor mediante la propiedad Value.

En el siguiente ejemplo se crea un element con un elemento secundario, usando el método para establecer el valor secundario:

```
// Creación del elemento sin atributo
XElement root = new XElement("Root");

// Se añaden los datos name/value
root.SetElementValue("Val1", 1);
root.SetElementValue("Val2", 2);
root.SetElementValue("Val3", 3);
Console.WriteLine(root);

// Modifica el name/value de Val2
root.SetElementValue("Val2", 102);
Console.WriteLine(root);

// Elimina el element si el valor es null
root.SetElementValue("Val3", null);
Console.WriteLine(root);
```

RECUERDA

Los atributos en XML son valores que se asocian a los elementos para dar una información adicional.

5. MANIPULACIÓN DE ELEMENTOS

Un paso más en el uso de los distintos elementos dentro de un documento XML, lo vas a encontrar a la hora de manipularlos para la consecución de las distintas acciones requeridas en la implementación del propio desarrollo web que estás realizando, como puede ser el caso, de la carga de archivos.

Al hablar de manipulación de elementos en XML, podemos partir de las opciones de carga para usar literales XML desde un origen externo como puede ser un archivo, secuencia o texto, pudiendo ser manipulado mediante lo que se conoce como LINQ (Language Integrated Query) y realizar la correspondiente consulta.

La carga de XML podemos realizarla desde:

1.- Un archivo

Permite rellenar un literal XML igual que si se tratara de un objeto tipo XElement o XDocument, mediante el método "load" que deberá contar como entrada, con una ruta de acceso bien a un archivo, secuencia de texto o XML.

A modo de ejemplo, podemos ver cómo sería el código fuente en el caso de hacerlo desde un archivo utilizando el método "Load (String)" con la intención de rellenar un objeto del tipo XDocument con XML, sería:

```
Dim books =
   XDocument.Load(My.Application.Info.DirectoryPath &
                          "\..\..\Data\books.xml")
Console.WriteLine(books)
```

2.- Una cadena

En este caso se utilizará el método "Parse", si es un objeto tipo XDocumento de XML el que queremos rellenar, tendríamos el siguiente código:

```
Dim xmlString = "<Book id=""bk102"">" & vbCrLf &
                          "  <Author>Garcia,
Debra</Author>" & vbCrLf &
                          "  <Title>Writing Code</Title>"
& vbCrLf &
                         "  <Price>5.95</Price>" & vbCrLf
&
                         "</Book>"
Dim xmlElem = XElement.Parse(xmlString)
Console.WriteLine(xmlElem)
```

3.- Una secuencia

Como en los casos anteriores, disponemos del método “Load”, pero también existe el método “XNode.ReadFrom” que nos permite rellenar un objeto del tipo XDocument con XML, tal y como vemos en el siguiente ejemplo:

```
Dim reader =

System.Xml.XmlReader.Create(My.Application.Info.Directory
Path &
                                "\..\..\Data\books.xml")
reader.MoveToContent()
Dim inputXml = XDocument.ReadFrom(reader)
Console.WriteLine(inputXml)
```

RESUMEN FINAL

En esta unidad se ha tratado el tema de manipulación de documentos web, tratando, en primer lugar, los lenguajes de script de cliente, su concepto, características y algunos de los más representativos como son JavaScript, Python o PHP.

Dada la importancia y su gran aceptación en la creación de documentos web, nos hemos centrado en el metalenguaje XML, su concepto y características, sirviéndonos de base para el desarrollo de los siguientes apartados de esta unidad relativos a los elementos que contiene un documento XML, con o sin atributos, así como los comentarios y esos posibles atributos que pueden acompañar a los elementos.

Importante conocer cómo crear o modificar los elementos y atributos, incluyendo el concepto de DTD, como documento que nos servirá para esa creación o modificación, ya sea de modo interno o externo.

El resto de acciones con elementos se han tratado tanto en el apartado dedicado a su eliminación, con los métodos de carga como “Remove”, y las opciones de utilización desde sentencia o nodo.

Para finalizar, se ha desarrollado los elementos y su manipulación, tanto para XElement como XDocument, y sus distintas posibilidades desde archivo, cadena o secuencia.