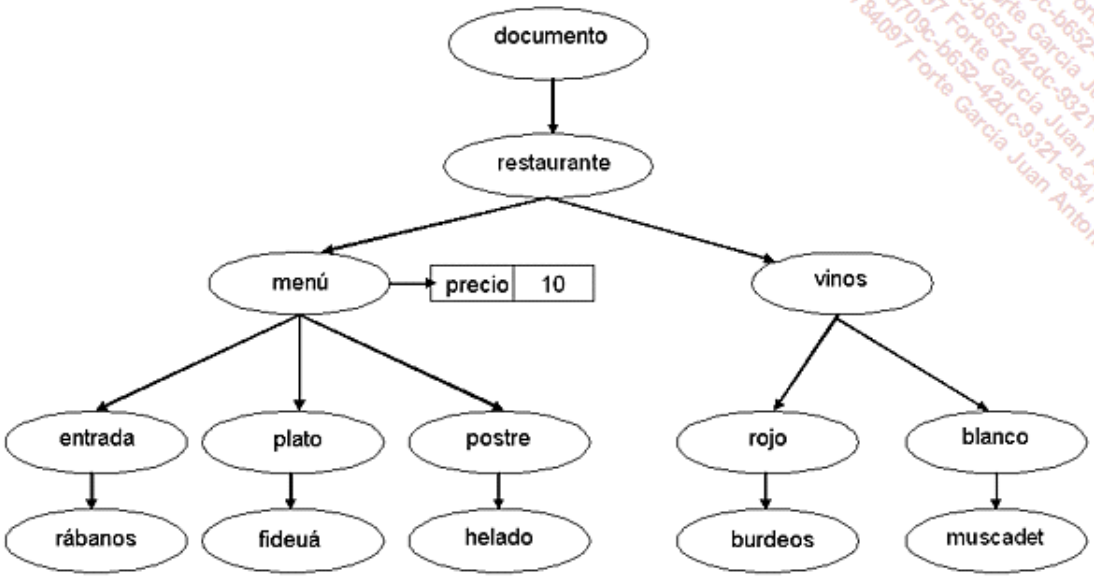
Manejo de un documento XML

El manejo de un documento XML en una aplicación C# viene facilitada por la utilización de DOM (*Document Object Model*). El DOM le permite leer, trabajar y modificar un documento XML por programa. Este último gobierna la representación en memoria de los datos XML, aunque los verdaderos datos XML estén almacenados de manera lineal cuando se encuentran en un archivo o provienen de otro objeto.

Por ejemplo, el siguiente documento:

```
<?xml version="1.0"?>
  <restaurante>
    <menu precio="10">
         <entrada>rábanos</entrada>
         <plato>fideuá</plato>
         <postre>helado</postre>
    </menu>
    <vinos>
          <tinto>burdeos</tinto>
          <blanco>muscadet</blanco>
    </vinos>
  </restaurante>
```

Se representa con esta forma en memoria en una aplicación:



En la estructura de un documento XML, cada círculo de esta ilustración representa un nodo, llamado objeto `XmlNode`, que es el objeto básico del árbol DOM. La clase `XmlDocument` se encarga de los métodos destinados a ejecutar las operaciones sobre el documento en su conjunto, por ejemplo para cargarlo en memoria o grabarlo en forma de archivo. Los objetos `XmlNode` comportan un conjunto de métodos y de propiedades, así como características básicas bien definidas. A continuación, presentamos algunas de estas características:

- Un nodo sólo puede poseer un nodo padre, que es el nodo situado justo encima de él.
- El único nodo que no tiene padre es la raíz del documento, ya que se trata del nodo de primer nivel que contiene el propio documento y los fragmentos de documento.
- La mayoría de los nodos pueden comportar varios nodos hijos, que son los nodos situados directamente bajo ellos.
- Los nodos situados en el mismo nivel, representados en el diagrama por los nodos menú y vinos, son nodos hermanos.

Una de las características del DOM es su manera de gestionar los atributos. Los atributos no son nodos que formen parte de las relaciones padre-hijo ni hermano. Se consideran una propiedad del nodo y están formados por un par, compuesto de un nombre y de un valor.

En nuestro ejemplo, precio="10" asociado al elemento menú, la palabra precio corresponde al nombre y el valor del atributo precio es 10. Para extraer el atributo precio="10" del nodo menú, se llama al método `GetAttribute` cuando el cursor se encuentra en el nodo menú.

Para los ejemplos siguientes, utilizaremos este documento XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<restaurante>
  <menu tipo="gastronomico">
    <entradas>
      <nombre calorias="50">rábanos</nombre>
      <nombre calorias="300">pasta</nombre>
      <nombre calorias="350">salchichón</nombre>
    </entradas>
    <platos>
      <nombre calorias="1000">paella</nombre>
      <nombre calorias="2000">fideuá</nombre>
      <nombre calorias="1700">cuscús</nombre>
    </platos>
    <quesos>
      <nombre calorias="240">manchego</nombre>
      <nombre calorias="300">tetilla</nombre>
      <nombre calorias="120">cabrales</nombre>
    </quesos>
    <postres>
      <nombre calorias="340" sabor="chocolate">helado</nombre>
      <nombre calorias="250" frutas="manzanas">tarta</nombre>
      <nombre calorias="400">crema catalana</nombre>
    </postres>
  </menu>
  <menu tipo="economico">
    <entradas>
      <nombre calorias="50">pan</nombre>
    </entradas>
    <platos>
      <nombre calorias="1700">jamón</nombre>
    </platos>
    <quesos>
      <nombre caloria="240">manchego</nombre>
    </quesos>
    <postres>
      <nombre calorias="340" sabor="fresa">helado</nombre>
    </postres>
  </menu>
</restaurante>
```

1. Utilización de DOM

La primera etapa durante la utilización de DOM consiste en cargar el documento XML en un árbol de nodos DOM. Para ello, debe declarar un objeto `XmlDocument` y luego utilizar el método `Load` con la finalidad de rellenar este objeto a partir

de un archivo XML.

```
XmlDocument doc;
    doc = new XmlDocument();
    doc.Load("restaurante.xml");
```

También es posible cargar datos XML a partir de una cadena de caracteres. En este caso, debe utilizar el método `LoadXML` y facilitar la cadena de caracteres que contiene los datos XML.

Una vez los datos XML estén cargados en el árbol, puede localizar nodos particulares con el fin de someterlos a operaciones de tratamiento o modificación. El método `GetElementsByTagName` permite obtener un objeto `XmlNodeList`, que contiene los nodos afectados. Entonces puede obtener los atributos del nodo usando la propiedad `Attributes` o comprobar si posee nodos hijos con la propiedad `HasChildNodes`. Si es el caso, tiene acceso a estos nodos a través de la propiedad `ChildNodes` en forma de un objeto `XmlNodeList`.

El siguiente ejemplo busca los nodos menú en el árbol y visualiza el atributo `type`.

```
XmlNodeList menus;
    menus=doc.GetElementsByTagName("menu");
    foreach( XmlNode unMenu in menus)
    {
        Console.WriteLine(unMenu.Attributes["type"].Value);
    }
```

También se pueden modificar las características de los nodos añadiéndoles un atributo. Los nodos pueden recibir, por ejemplo, un atributo **precio**.

```
menus = doc.GetElementsByTagName("menu");
    XmlAttribute att;
    foreach ( XmlNode unMenu in menus)
    {
        if (unMenu.Attributes["type"].Value == "gastronomico")
        {
            att = doc.CreateAttribute("precio");
            att.Value = "50€";
            unMenu.Attributes.Append(att);
        }
        if (unMenu.Attributes["type"].Value == "economico")
        {
            att = doc.CreateAttribute("precio");
            att.Value = "15€";
            unMenu.Attributes.Append(att);
        }
    }
```

También es posible añadir nodos hijos a nodos que existen en el árbol, creando instancias de la clase `XmlNode` y uniéndolos a su nodo padre. El siguiente ejemplo añade un **digestivo** al menú **gastronomico**.

```
menus = doc.GetElementsByTagName("menu");
    XmlAttribute att;
    foreach ( XmlElement unMenu in menus)
    {
     if (unMenu.Attributes["tipo"].Value == "gastronomico")
     {
        XmlNode n1;
        XmlNode n2;
        XmlNode n3;
```

```
        n1 = doc.CreateNode(XmlNodeType.Element, "digestivo", "");
        n2 = doc.CreateNode(XmlNodeType.Element, "nombre", "");
        n3 = doc.CreateNode(XmlNodeType.Text, "", "");
        n3.Value = "Cognac";
        n2.AppendChild(n3);
        n1.AppendChild(n2);
        unMenu.AppendChild(n1);
     }
    }
```

Después de la ejecución de los dos ejemplos anteriores, el documento XML debe presentar la siguiente forma:

```
<?xml version="1.0" encoding="Windows-1252"?>
<restaurante>
  <menu tipo="gastronomico" precio="50€">
    <entradas>
      <nombre calorias="50">rábanos</nombre>
      <nombre calorias="300">pasta</nombre>
      <nombre calorias="350">salchichón</nombre>
    </entradas>
    <platos> <nombre calorias="1000">paella</nombre>
      <nombre calorias="2000">fideuá</nombre>
      <nombre calorias="1700">cuscús</nombre>
    </platos>
    <quesos>
      <nombre calorias="240">manchego</nombre>
      <nombre calorias="300">tetilla</nombre>
      <nombre calorias="120">cabrales</nombre>
    </quesos>
    <postres>
      <nombre calorias="340" sabor="chocolate">helado</nombre>
      <nombre calorias="250" frutas="manzanas">tarta</nombre>
      <nombre calorias="400">crema catalana</nombre>
    </postres>
    <digestivo>
      <nombre>Cognac</nombre>
    </digestivo>
  </menu>
  <menu tipo="economico" precio="15€">
    <entradas>
      <nombre calorias="50">pan</nombre>
    </entradas>
    <platos>
      <nombre calorias="1700">jamón</nombre>
    </platos>
    <quesos>
      <nombre calorias="240">manchego</nombre>
    </quesos>
    <postres>
      <nombre calorias="340" sabor="fresa">helado</nombre>
    </postres>
  </menu>
</restaurante>
```

En realidad, sólo se modifica la representación en memoria del documento XML. Si desea conservar las modificaciones,

debe registrar el documento en un archivo para asegurar la persistencia de los datos. Para ello, debe utilizar el método save de la clase `XmlDocument`, y facilitar el nombre del archivo en el cual desea efectuar la copia de seguridad.

```
doc.Save("restaurante2.xml");
```

2. Utilización de XPath

El principal objetivo de XPath consiste en definir la manera de dirigirse a partes de un documento XML. El nombre XPath viene de la utilización de una escritura de tipo «path», como en los shells DOS y UNIX. El objetivo consiste en desplazarse en el interior de la estructura jerárquica de un documento XML como si se tratase de un árbol de directorios. Para darse cuenta del interés de XPath, podríamos decir que es el equivalente del lenguaje SQL para un documento XML. La comparación debe detenerse aquí, ya que ¡la sintaxis de ambas no tiene nada que ver entre sí!

a. Búsqueda en un documento XML

Para buscar un elemento en un documento XML, la primera etapa consiste en crear una instancia de la clase `XPathNavigator`. Esta instancia de clase debe conocer el documento en el cual tendrá que hacer búsquedas. Por eso, el propio documento, por medio del método `CreateNavigator`, va a facilitar esta instancia de clase.

```
    XPathNavigator navegador;
    navegador = doc.CreateNavigator();
```

A partir de esta instancia, vamos a poder iniciar búsquedas en el documento usando el método `Select`. Este método utiliza como parámetro una cadena de caracteres que contiene la ruta `XPath` de búsqueda. Después de la ejecución, obtenemos un objeto `XPathNodeIterator`, que pemite recorrer la lista de los nodos encontrados.

El siguiente ejemplo busca en el documento **restaurante.xml** las entradas disponibles en los diferentes menús:

```
XmlDocument document = new XmlDocument();
doc.Load("restaurante.xml");
XPathNavigator navegador = doc.CreateNavigator();
XPathNodeIterator nodos = navegador.Select("/restaurante/menu/entradas");
while (nodos.MoveNext())
{
    Console.WriteLine(nodos.Current.OuterXml);
    Console.WriteLine();
}
```

Obtenemos el siguiente resultado:

```
<entradas>
  <nombre calorias="50">rábanos</nombre>
  <nombre calorias="300">pasta</nombre>
  <nombre calorias="350">salchichón</nombre>
</entradas>
<entradas>
  <nombre calorias="50">pan</nombre>
</entradas>
```

También es posible añadir a la petición `XPath` los criterios de selección sobre el valor de ciertos atributos.

El siguiente ejemplo busca los postres del menú **gastronomico** con menos de **350** calorías.

```
XmlDocument doc = new XmlDocument();
 doc.Load("restaurante.xml");
```

```
XPathNavigator navegador = doc.CreateNavigator();
XPathNodeIterator nodos = navegador.Select(
   "/restaurante/menu[@type='gastronomico']/postres/nombre[@calorias<350]");
while (nodos.MoveNext())
{
    Console.WriteLine(nodos.Current.Value);
    Console.WriteLine();
}
```

b. Modificación de los datos de un documento XML

Después de haber encontrado un elemento en el árbol de un documento, es posible modificar su valor.

El siguiente ejemplo disminuye un **50%** las calorías de cada postre del menú **gastronomico**.

```
doc = new XmlDocument();
doc.Load("restaurante.xml");
XPathNavigator navegador = doc.CreateNavigator();
XPathNodeIterator nodos = navegador.Select(
        "/restaurante/menu[@tipo='gastronomico']/postres/nombre");
while (nodos.MoveNext())
 {
    nodos.Current.MoveToAttribute("calorias", "");
    nodos.Current.SetValue(String.Format("{0:####}",
                          (double.Parse(nodos.Current.Value) * 0.5)));
 }
 doc.Save("restaurante.xml");
```

A continuación presentamos el contenido del archivo después de la ejecución del código anterior.

```
<?xml version="1.0" encoding="utf-8" ?>
<restaurante>
  <menu tipo="gastronomico">
    <entradas>
      <nombre calorias="50">rábanos</nombre>
      <nombre calorias="300">pasta</nombre>
      <nombre calorias="350">salchichón</nombre>
    </entradas>
    <platos>
      <nombre calorias="1000">chucrut</nombre>
      <nombre calorias="2000">cocido</nombre>
      <nombre calorias="1700">cuscús</nombre>
    </platos>
    <quesos>
      <nombre calorias="240">manchego</nombre>
      <nombre calorias="300">tetilla</nombre>
      <nombre calorias="120">cabrales</nombre>
    </quesos>
    <postres>
      <nombre calorias="170" sabor="chocolate">helado</nombre>
      <nombre calorias="125" frutas="manzanas">tarta</nombre>
      <nombre calorias="200">crema catalana</nombre>
    </postres>
  </menu>
  <menu type="economico">
```

```
    <entradas>
      <nombre calorias="50">pan</nombre>
    </entradas>
    <platos>
      <nombre calorias="1700">jamón</nombre>
    </platos>
    <quesos>
      <nombre calorias="240">manchego</nombre>
    </quesos>
    <postres>
      <nombre calorias="340" sabor="fresa">helado</nombre>
    </postres>
  </menu>
</restaurante>
```

c. Añadir un nodo a un documento XML

Después de buscar un nodo en un documento, es posible añadirle nodos hijos y nodos hermanos. Los métodos `InsertAfter` y `InsertBefore` añaden un nodo hermano después o antes del nodo actual. El método `AppendChild` añade un nodo hijo al nodo actual.

El siguiente ejemplo añade un nuevo postre al menú **gastronomico**.

```
XmlDocument doc;
    doc = new XmlDocument();
    doc.Load("restaurante.xml");
    XPathNavigator navegador = doc.CreateNavigator();
    XPathNodeIterator nodos =   navegador.Select(
             "/restaurante/menu[@tipo='gastronomico']/postres");
    nodos.MoveNext();
    nodos.Current.AppendChild("<nombre calorias='800'>crepes</nombre>");
    doc.Save("restaurante.xml");
```

Después de la ejecución de este código, el documento se convierte en:

```
<?xml version="1.0" encoding="utf-8" ?>
<restaurante>
 <menu tipo="gastronomico">
  <entradas>
   <nombre calorias="50">rábanos</nombre>
   <nombre calorias="300">pasta</nombre>
   <nombre calorias="350">salchichón</nombre>
  </entradas>
  <platos>
   <nombre calorias="1000">paella</nombre>
   <nombre calorias="2000">cocido</nombre>
   <nombre calorias="1700">cuscús</nombre>
  </platos>
  <quesos>
   <nombre calorias="240">manchego</nombre>
   <nombre calorias="300">tetilla</nombre>
   <nombre calorias="120">cabrales</nombre>
  </quesos>
  <postres>
   <nombre calorias="340" sabor="chocolate">helado</nombre>
```

```
   <nombre calorias="250" frutas="manzanas">tarta</nombre>
   <nombre calorias="400">crema catalana</nombre>
   <nombre calorias="800">crepes</nombre>
  </postres>
 </menu>
 <menu tipo="economico">
  <entradas>
   <nombre calorias="50">pan</nombre>
  </entradas>
  <platos>
    <nombre calorias="1700">jamón</nombre>
  </platos>
  <quesos>
   <nombre calorias="240">manchego</nombre>
  </quesos>
  <postres>
   <nombre calorias="340" sabor="fresa">helado</nombre>
  </postres>
 </menu>
</restaurante>
```