实 验 五　XML 文档操作

5.1　实验目的

XML 在程序设计中应用场合广泛，本实验介绍在.NET 平台中基本的 XML 文件的创建和读写，XmlDocument 类与 XmlNode 类编辑 xml 节点的方法，还有使用 Schema 对 xml 文件数据验证的方法。

5.2　XML 文档介绍

网页语言 HTML 称为静态网页是一种标记语言，它的标签数量固定，用法固定，每种标签预定义了内容的显示格式，由浏览器显示，重在数据的显示模式。XML 也是一种标记语言，XML 没有对标签进行预定义，而由用户自定义和解析称作自描述性；它用来存储结构化数据，偏重数据本身。XML 标签可任意扩展，通过 DTD 或者 Schema 可将 XML 文档进行内容规定。XML 文档具有较严格的结构性，它以节点为基本元素，所有结点形成结点树。根元素是所有其他元素的父元素，根元素没有父级的节点，其它节点都有一个父节点，大多数节点可以有多个子节点。

在多数语言操作 XML 文档时，有两种主要方式，SAX 方式和 DOM 方式，SAX 方式是以流并且只读的方式处理 XML 节点信息，而 DOM 预先将文档全部读入内存，提供随机访问方式，更灵活。SAX 方式提供非缓存的只进、只读访问，在.NET 平台中提供与 SAX 类似的 XmlReader 类，它是只向前读的 XML 读取器，这意味着使用 XmlReader 无法编辑属性值或元素内容，也无法插入和移除节点。XmlDocument 类则以 DOM 方式操作 XML，它与其相关类可以用来构造 XML 文档、加载数据，修改数据。本实验主要介绍 DOM 方式操作 XML。

5.2.1　XML 文档对象模型 DOM

XML 文档对象模型 (DOM) 类是 XML 文档的内存中表示形式，编辑是 DOM 的主要功能。实际的 XML 数据在文件中时或从另一个对象传入时以线性方式存储，在 XML 数据读入内存后，采用的是结构化表示方法。看下面的 XML 文档内容：

```
<?xml version='1.0'?>
<books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
```

```
    </book>
    <pubinfo>
      <publisher>MSPress</publisher>
      <state>WA</state>
    </pubinfo>
  </books>
```

其结构化的内存构造表示如图5-1 在 XML 文档结构中，此图中的每个圆圈表示一个节点，

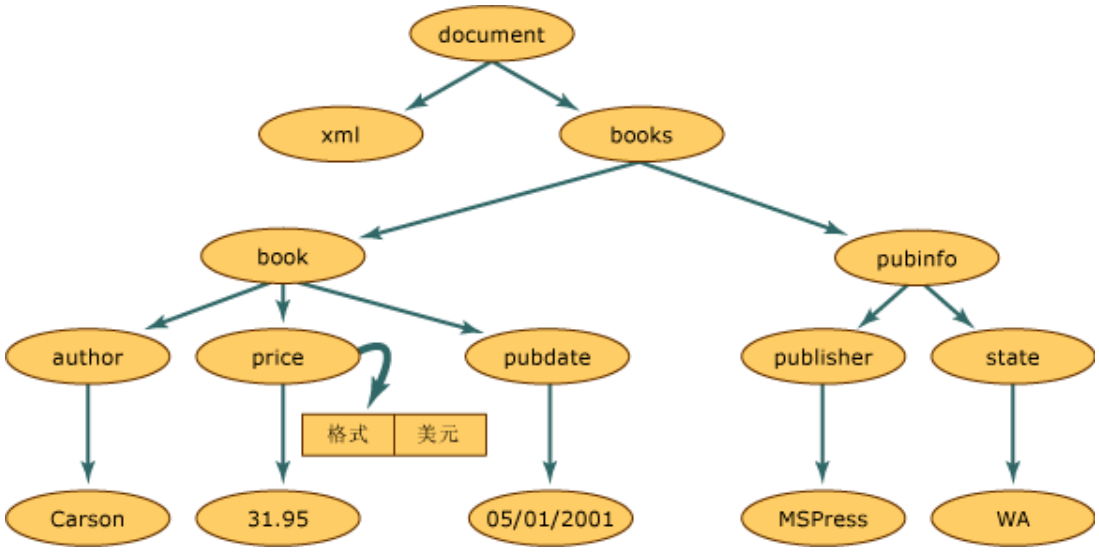

图 5-1 book.xml 的 DOM 对象模型

在.NET 中称为 XmlNode 对象，它是 DOM 树中的基本对象，xml 文档中节点在 DOM 中以树的结构形式组织在一起。

5.2.2 操作 XML 文件的类

.NET 平台提供了操作 XML 文件的类，XmlReader 类用于读入 XML 文档，它也可用于创建 XML 文档，XmlWriter 用于向 XML 文档写入内容，XmlNode 类用于 XML 节点的操作。XmlDocument 类（扩展 XmlNode）支持用于对整个文档执行操作（例如，将文档加载到内存中或将 XML 保存到文件中）的 DOM 方法。XmlDocument 类是最重要的操作 XML 文件的类，它表示整个 XML 文档中的内存对象，操作 XML 必须先创建这个对象。XmlDocument 提供了查看和处理整个 XML 文档中的节点的方法。XmlNode 和 XmlDocument 通过方法和属性执行节点操作，包括访问和修改 DOM 特定的节点，如元素节点、实体引用节点等，检索节点包含的信息（如元素节点中的文本）或者整个节点。XmlNode 类表示 XML 中的单个节点，文档大部分的操作都是基于节点对象，这个类提供大量方法来操作 XML 中的节点。节点概念在 DOM 中非常重要，XML 对节点进行了分类，节点的类型规定了可执行的操作，表5-1列出 xml 文档中节点的类型。

Document 类型节点表示包容了所有的节点，DocumentFragment 类型节点包括若干个节点，Comment 类型节点是文档注释部分，图5-2指出了 Element、Atrribute、Text 类型节点的示意。

表 5-1 XML 节点类型

节点类型	类名	意义	子节点
Document	XmlDocument	文档根	有
DocumentFragment	XmlDocumentFragment	片段	有
EntityReference	XmlEntityReference	实体引用	有
Element	XmlElement	表示元素节点	有
Attribute	XmlAttribute	元素的属性	有
Declaration	XmlDeclaration	XML 声明	无
Notation	XmlNotation	声明中的表示法	无
Entity	XmlEntity	实体声明	无
CDATA	XmlCDataSection	CDATA 节	无
Comment	XmlComment	注释	无
ProcessingInstruction	XmlProcessingInstruction	处理指令	无
DocumentType	XmlDocumentType	文档类型声明	无

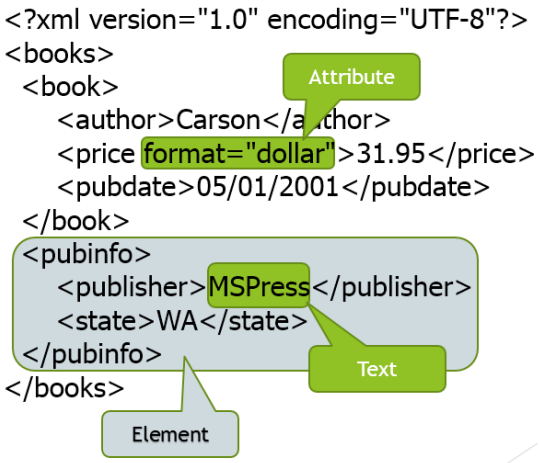

图 5-2 XML 文档中的节点类型

5.3 使用 DOM 操作 xml 文档

本节内容是以 DOM 方式操作 xml 文档，包括创建 xml 文件，显示 xml 的 DOM 树结构，修改 xml 文档中节点的值。

5.3.1 创建 xml 文件

打开 VS2013 工具，创建一个窗体应用程序，项目位置为 D:\jiao，名称为 xml，将其生成路径设为当前路径 (即.)。在窗体上放置 textBox 控件设其支持多行文本，添加 treeView 控件以及按钮，参考界面如图5-3：

在按钮创建 XML 文件的点击事件中加入下面的代码，它用于生成一个 XML 文档，并保存为 bookstore.xml 文件。

```
//创建一个 XML 文档
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.LoadXml("<bookstore/>");
```

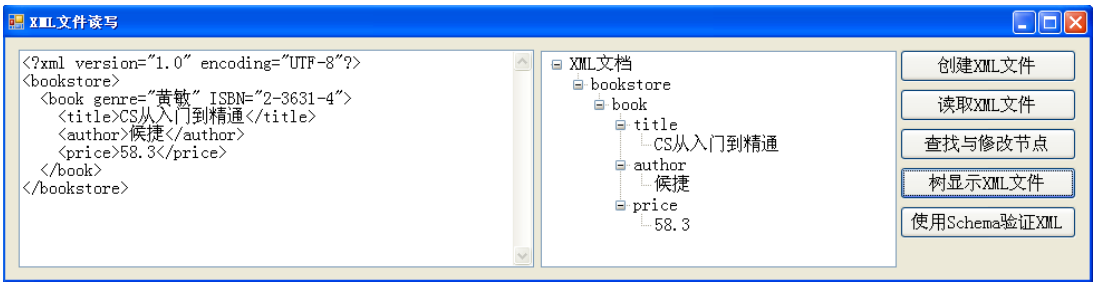

图 5-3 XML 操作参考界面

```
//创建 XML 版本声明.
XmlDeclaration xmldecl;
xmldecl = xmlDoc.CreateXmlDeclaration("1.0","UTF-8", null);
//根节点
XmlNode root = xmlDoc.DocumentElement;
xmlDoc.InsertBefore(xmldecl, root);
//创建一个节点
XmlElement xe1 = xmlDoc.CreateElement("book");
//设置该节点 genre 属性
xe1.SetAttribute("genre", " 黄敏");
//设置该节点 ISBN 属性
xe1.SetAttribute("ISBN", "2-3631-4");
XmlElement xesub1 = xmlDoc.CreateElement("title");
xesub1.InnerText = "CS 从入门到精通";//设置文本节点
xe1.AppendChild(xesub1);//添加到节点中
XmlElement xesub2 = xmlDoc.CreateElement("author");
xesub2.InnerText = " 候捷";
xe1.AppendChild(xesub2);
XmlElement xesub3 = xmlDoc.CreateElement("price");
xesub3.InnerText = "58.3";
xe1.AppendChild(xesub3);
root.AppendChild(xe1);//添加到节点中
xmlDoc.Save("bookstore.xml");
```

创建 xml 文档的程序使用了 XmlDocument 类的 DocumentElement 成员，它代表文档的根节点，XmlDeclaration 类代表 xml 文档的版本声明部分，通过 XmlDocument 类的 InsertBefore 方法，将声明插入到根节点前；XmlDocument 类的 CreateElement 用于创建一个普通的 XmlElement 节点对象，使用 XmlElement 的 AppendChild 方法添加节点的子节点对象；最后使用 XmlDocument 类的 Save 方法将生成的 xml 文档以给定的文件名进行保存。

5.3.2 显示 xml 文本内容

实现读取 XML 文件的按钮事件比较简单，窗体上放置一个 openFileDialog 控件，用户点击按钮后将 xml 文本内容显示在文本框中，并把 xml 文档名记录在全局变量中，按钮事件的参考代码如下：

```
openFileDialog1.InitialDirectory= Directory.GetCurrentDirectory();
openFileDialog1.Filter = "XML 文档 |*.xml";
StreamReader sr=null;
if(openFileDialog1.ShowDialog()==DialogResult.OK)
{
  xml_filename = openFileDialog1.FileName;
  sr=new StreamReader(xml_filename);
  textBox1.Text = sr.ReadToEnd();
  sr.Close();
}
```

5.3.3 显示 xml 节点的 DOM 树结构

DOM 方式将 XML 文档的节点元素构造成一根树，程序使用 TreeView 控件显示对应的 DOM 树，树是递归定义结构，要编写递归函数来构造树节点。先要获取文档的根节点，将根节点作为递归函数初始调用参数，获取文档的根节点参考代码如下：

```
XmlDocument doc = new XmlDocument();
if(File.Exists(xml_filename))
{
  doc.Load(xml_filename);
  treeView1.Nodes.Clear();
  XmlNode root = doc.DocumentElement;
  TreeNode node;
  node = treeView1.Nodes.Add("XML 文档");
  build_tree(root,node);
  treeView1.ExpandAll();
}
```

树视图控件结点构造过程由递归函数 build_tree 完成，这个函数的两个参数分别是 DOM 节点和树节点对象，通过遍历 XML 文档中的 DOM 节点，获得其子节点后建立树视图结点关系，以调用自身的方式构造子树。build_tree 函数参考代码如下：

```
private void build_tree(XmlNode xnode,TreeNode p_tnode)
{
  TreeNode node;
  switch (xnode.NodeType)
  {
    case XmlNodeType.Element:
```

```
        {
            node = p_tnode.Nodes.Add(xnode.Name);
            if (xnode.HasChildNodes)
            {
                for (int i = 0; i < xnode.ChildNodes.Count; i++)
                {
                    build_tree(xnode.ChildNodes[i], node);
                }
            }
            break;
        }
        case XmlNodeType.Text:
        {
            p_tnode.Nodes.Add(xnode.Value);
            break;
        }
    }
}
```

点击树节点后显示 XML 文档对应的 DOM 树，运行结果可参考图5-3。

5.3.4 编辑 xml 节点

DOM 能够以随机方式直接访问节点元素并编辑节点值，下面代码查找 xml 节点名，修改节点的属性以及节点文本值。

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load("bookstore.xml");
//获取 bookstore 节点的所有子节点
XmlNodeList nodeList = xmlDoc.SelectSingleNode("bookstore").ChildNodes;、、  //遍历子节点
foreach (XmlNode xn in nodeList)
{
    //将子节点类型转换为 XmlElement 类型
    XmlElement xe = (XmlElement)xn;
    //如果属性值匹配
    if (xe.GetAttribute("genre") == " 黄敏")
    {
        //修改属性为“update 黄敏”
        xe.SetAttribute("genre", "update 黄敏");
        //继续获取 xe 子节点的所有子节点
        XmlNodeList nls = xe.ChildNodes;
```

```
        foreach (XmlNode xn1 in nls)//遍历
        {
          XmlElement xe2 = (XmlElement)xn1;//转换类型
          //匹配节点名称
          if (xe2.Name == "author")
          {
            //进行修改
            xe2.InnerText = " 亚胜";
            break;//不再继续遍历
          }
        }
        break;
      }
    }
    xmlDoc.Save("bookstore.xml");//保存。
```

程序使用 XmlDocument 类的 SelectSingleNode 方法根据名称定位节点，这个方法只返回最先匹配的节点，如果在 DOM 节点树中有重名的节点，可使用 SelectNodes 方法，它将返回所有匹配的节点集合。

5.3.5 XML 其它方法

以 DOM 方式操作 xml 文档可通过 XmlNode 类对象和 XmlDocument 类对象实现，其它方法与上述操作 xml 的过程类似，读者可进一步参考 MSDN。在 DOM 节点之前添加节点可用下面的方法：

```
XmlNode refChild = node.ChildNodes[4];
//The reference is zero-based.
node.InsertBefore(newChild, refChild);
```

RemoveChild 方法移除 XML 文档对象模型 (DOM) 中的节点。

```
XmlNode root = doc.DocumentElement;
//Remove the title element.
root.RemoveChild(root.FirstChild);
```

移除 XML 文档对象模型 (DOM) 中的节点的属性：

```
XmlAttributeCollection attrColl = doc.DocumentElement.Attributes;
attrColl.Remove(attrColl["genre"]);
XmlElement root = doc.DocumentElement;
// Remove the genre attribute.
root.RemoveAttribute("genre");
```

5.4 使用 Schema 验证 xml 文件

XML Schema 与 DTD 一样，是对 XML 文档进行约束并确定其结构，包括对元素、属性、及数据类型的完整定义，及某个 XML 文档中所使用的元素、实体、元素的属性、元素与实体之间的关系的特殊约束。用户可根据需要制定不同的 Schema 内来约束不同场合下的 xml 数据。一个 Schema 文件本身也是一个 xml 文档，Schema 文档后缀为 xsd，Schema 定义了多种数据类型，像在其他编程语言那样例如整型、字符型、浮点型、布尔型、日期型等。使用 schema 可以控制元素的值类型，元素之间的关系，节点成员组成等等。在此不对 Schema 语法及如何生成 Schema 作深入探讨，读者可参考相关技术文档。本小节通过两个文件来说明如何使用 Schema 来验证 xml 数据的有效性，这反映出 xml 文档在数据表示方面的强大功能。

项目提供三个文档，booksSchemaGood.xml，booksSchemaFail.xml 是待验证的 xml 数据，它们的内容是包含了一定结构的数据记录，books.xsd 是对数据的约束，它规定了数据记录的格式，如节点的成员及其名称，字段的值类型。booksSchemaGood.xml 是格式良好的 xml 文档，在此仅作为对比功能，因此它的验证是没有问题的，booksSchemaFail.xml 的数据记录有部分不符定义的内容，在使用 Schema 验证时会被检测出格式不正确的地方。

使用.NET 平台类对读入的 xml 文档进行验证时，使用了事件回调机制，当发生错语时，将调用回调函数，编写方法如下：

```
private static void ValidationCallBack(object sender, ValidationEventArgs e)
{
  MessageBox.Show(String.Format("Validation Error: {0}", e.Message));
}
```

其中事件的名称可以自定义，但是参数列表是固定不变的，上述函数在错误发生仅给出错误的信息。使用 Schema 对 xml 文档进行数据验证的事件代码可参考如下：

```
XmlSchemaSet sc = new XmlSchemaSet();
sc.Add("urn:bookstore-schema", "books.xsd");
XmlReaderSettings settings = new XmlReaderSettings();
//ConformanceLevel = ConformanceLevel.Document
settings.ValidationType = ValidationType.Schema;
settings.Schemas = sc;
settings.ValidationEventHandler += new ValidationEventHandler(ValidationCallBack);
XmlReader reader = XmlReader.Create(xml_filename, settings);
// Parse the file.
while (reader.Read()) ;
```

在读入 xml 文档前，需要创建 XmlReaderSettings 类对象，这个对象设置了验证类型为 Schema，Schema 文件名为 books.xsd，并将回调事件进行绑定；使用 XmlReader 对象读入指定的 xml 文档，在读入过程中给出所有出错的节点信息。

```
<?xml version='1.0'?>
<bookstore xmlns="urn:bookstore-schema">
<book>
  <author>
```

```
    <first-name>Benjamin</first-name>
    <last-name>Franklin</last-name>
    </author>
  </book>
  <book genre="novel">
    <title>The Confidence Man</title>
    <author>
    <first-name>Herman</first-name>
    <last-name>Melville</last-name>
    </author>
    <price>11.99</price>
  </book>
  <book genre="philosophy">
    <title>The Gorgias</title>
    <author>
    <name>Plato</name>
    </author>
    <price>9.99</price>
  </book>
  </bookstore>
```

5.5 作业

1. 添加树控件事件代码使得用户可以添加或删除树节点。
2. 编写将树控件中节点写入 XML 文件功能。