# XML(可扩展标记语言)

XML: extensible Markup Language

## 什么是标记语言？什么是标记

标记（markup）：文档中任何不想被打印输出的部分（不是真正的文档的内容，就好像注解，与文章实际内容无关），标记就是对内容的注解。

标记的作用：传递了文档本身以外的额外信息。比如:标记文档的某部分该如何显示，某部分是什么意思等。标记只是为了说明数据的含义。

常见的标记语言：SGML、HTMl、XML

## HTML与XML的区别与联系

XML用来存储数据，html用来展示数据

## XML语法特点

1. 严格区分大小写
2. 有且只有一个根节点
3. 有开始标签就必须有结束标签，至少得是自关闭标签
4. 属性必须使用双引号
5. 单纯的xml文档，没有预定义标签
6. 注释与html一样<!-- -->
7. <![CDATA[….]]>: CDATA的作用：把其中的所有文字当作纯文本，不识别标签。如：<![CDATA[a的值: 1<a>3]]>

## XML的主要作用

XML容易读懂，格式标准化语言都内置了XML分析引擎，不用单独进行文件分析引擎的编写。

XML是用一种格式化的方式来存储数据，并且是文本文件。任何时候都能够用记事本打开，与word、excel不一样。

XML便于数据交换。纯文本的存储方式可以随时随地的对其进行解读。其优点也是其缺点，那就是不安全。一般用来进行数据的中转（如两个数据库互相不兼容，但是都兼容xml就可以通过xml中转）、导入导出操作。

XML常用来做程序的配置文件。

# XML文档读写

## 读写XML的技术

读写XML有很多技术，主要有以下几种：

1. Dom方式: 也就时文档对象模型，将整个XML加载到内存中，然后操作。在.net中用的XmlDocument和XDocument。
2. 事件驱动方式：java中是Sax，.net中用XmlReader、XmlWriter。
3. Xml序列化：XmlSerializer，需要先定义类
4. Linq to XML：在System.Xml.Linq命名空间下，用到的就是XDocument，Xelement。

## XDocument写入Xml文档

|  |
| --- |
| using System;  using System.Xml.Linq;//引入命名空间  namespace \_01\_XDocument写入XML文档  {  class Program  {  static void Main(string[] args)  {  //1. 创建一个空文档对象  XDocument document = new XDocument();  //2. 创建一个文档声明  XDeclaration xDec = new XDeclaration("1.0", "utf-8", "yes");  //2.1 用Declaration设置文档声明  document.Declaration = xDec;  //3.创建一个根节点  XElement rootElement = new XElement("Order");  //3.1 根节点添加到文档中  document.Add(rootElement);  //4. 创建一个CustomerName节点。  XElement customerName = new XElement("CustomerName");  //4.1 节点中内容为郝玉锋  customerName.SetValue("郝玉锋"); ;  //4.2 添加CustomerName到根节点下。  rootElement.Add(customerName);  // 5， 创建一个OrderNumber节点.  rootElement.SetElementValue("OrderNumber", "JD20150314");  //6. 创建一个item节点  XElement item = new XElement("Item");  //6.1 添加到根节点下。  rootElement.Add(item);  //7创建一个orderitem  XElement orderitem = new XElement("OrderItem");  //7.1 设置属性  orderitem.SetAttributeValue("Name", "电脑");  orderitem.SetAttributeValue("Count", 30);  //7.2添加到orderitem节点  item.Add(orderitem);  //8把对象写入到Order.xml文档中  document.Save("Order.xml");  }  }  } |

### 创建xml文档的步骤

创建xml文档的步骤主要分为：

1. 创建文档对象
2. 创建并设置文档声明
3. 创建并添加根节点到文档中。
4. 创建节点并添加到根节点下
5. 把文档对象写入到xml文件中。

#### 创建文档对象

|  |
| --- |
| //1. 创建一个空文档对象  XDocument document = new XDocument(); |

#### 创建并设置文档声明

|  |
| --- |
| //2. 创建一个文档声明  XDeclaration xDec = new XDeclaration("1.0", "utf-8", "yes");  //2.1 用Declaration设置文档声明  document.Declaration = xDec; |

#### 创建并添加根节点到文档中

|  |
| --- |
| //3.创建一个根节点  XElement rootElement = new XElement("Order");  //3.1 根节点添加到文档中  document.Add(rootElement); |

#### 创建节点并添加到根节点下

创建于添加节点有两种方式。第一种方式有返回值，可以通过返回值为其设置属性和添加子节点。

|  |
| --- |
| //4. 创建一个CustomerName节点。  XElement customerName = new XElement("CustomerName");  //4.1 节点中内容为郝玉锋  customerName.SetValue("郝玉锋") ;  customerName.SetAttributeValue("id", "JD200");  //4.2 添加CustomerName到根节点下。  rootElement.Add(customerName); |

第二种方式比较快捷但是没有返回值，没有办法设置属性，也无法为其添加子节点

|  |
| --- |
| // 5， 创建一个OrderNumber节点.  rootElement.SetElementValue("OrderNumber", "JD20150314"); |

#### 把文档对象写入到xml文件中

|  |
| --- |
| //8把对象写入到Order.xml文档中  document.Save("Order.xml"); |

## 集合中对象属性写入到Xml文档

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Xml.Linq;  namespace \_02\_集合中对象写入到xml文件  {  class Program  {  static void Main(string[] args)  {  List<Person> list = new List<Person>(){  new Person(){Name="秦始皇",Age=18,Email="hyfdbd@qq.com"},  new Person(){Name="汉武帝",Age=18,Email="hwd@qq.com"},  new Person(){Name="唐太宗",Age=18,Email="sgz@qq.com"},  };  //1. 创建xml文档对象  XDocument document = new XDocument();  //2. 创建于添加文档声明  XDeclaration xDec = new XDeclaration("1.0", "utf-8", "no");  document.Declaration = xDec;  //3. 创建并添加根节点到文档对象  XElement rootElement = new XElement("List");  document.Add(rootElement);  //4.遍历集合中元素并添加到根节点下  foreach (Person item in list)  {  XElement KingElement = new XElement("King");  rootElement.Add(KingElement);  KingElement.SetElementValue("Name", item.Name);  KingElement.SetElementValue("Age", item.Age);  KingElement.SetElementValue("Email", item.Email);  }  //5. 写入到xml文件  document.Save("King.xml");  Console.WriteLine("ok");  Console.ReadLine();  }  }  class Person  {  public string Name { get; set; }  public int Age { get; set; }  public string Email { get; set; }  }  } |

## 读取xml文件

xml文件的内容如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="no"?>  <List>  <King time ="公元前259年-公元前210年">  <Name>秦始皇</Name>  <Age>18</Age>  <Email>hyfdbd@qq.com</Email>  </King>  <King time="公元前156年]－公元前87年">  <Name>汉武帝</Name>  <Age>18</Age>  <Email>hwd@qq.com</Email>  </King>  <King time="公元598年-公元649年">  <Name>唐太宗</Name>  <Age>18</Age>  <Email>sgz@qq.com</Email>  </King>  </List> |

下面的是读取xml文件的方法:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Xml.Linq;  namespace \_03\_读取xml  {  class Program  {  static void Main(string[] args)  { //1.加载xml文件到Dom对象中。  XDocument document = XDocument.Load("King.xml");  //2. 获取根节点  XElement rootElement = document.Root;  //3. 打印根节点名称  Console.WriteLine(rootElement.Name.ToString());  Console.WriteLine("+++++++++++++++++++++++++++++++++++++++++++++++++");  PrintAttributes(rootElement);  //4.获取当前节点下所有名字为King的子节点。如果不传参数，则获取所有节点。  //返回值类型为 IEnumerable<XElement>。  IEnumerable<XElement> elements = rootElement.Elements("King");  //遍历每一个元素。  foreach (XElement item in elements)  {  Console.WriteLine("节点： {0}", item.Name);  Console.WriteLine("---------------------------------------------");  //打印属性的方法  PrintAttributes(item);  //获取子元素  IEnumerable<XElement> subElements = item.Elements();  //遍历子元素  foreach (XElement subElement in subElements)  { //打印子元素的名称和值  Console.WriteLine("子节点: {0}={1}", subElement.Name, subElement.Value);  }  Console.WriteLine("=============================================");  }  Console.ReadLine();  }  /// <summary>  /// 打印属性的方法  /// </summary>  /// <param name="item"></param>  private static void PrintAttributes(XElement item)  {  IEnumerable<XAttribute> attrs = item.Attributes();  foreach (XAttribute attr in attrs)  {  Console.WriteLine("属性：{0} = {1}", attr.Name, attr.Value);  Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");  }  }  }  } |

### 案例：读取订单

xml格式如下：

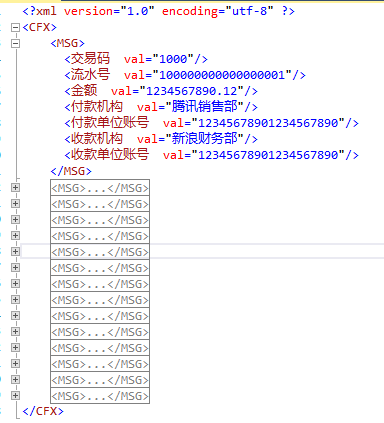
|  |
| --- |
| <?xml version="1.0" encoding="utf-8" standalone="yes"?>  <Order>  <CustomerName id="JD200">郝玉锋</CustomerName>  <OrderNumber>JD20150314</OrderNumber>  <Item>  <OrderItem Name="电脑" Count="30" />  <OrderItem Name="电视" Count="10" />  <OrderItem Name="冰箱" Count="2" />  </Item>  <CustomerName id="JD200">秦始皇</CustomerName>  <OrderNumber>JD2011014</OrderNumber>  <Item>  <OrderItem Name="战马" Count="30" />  <OrderItem Name="刀" Count="10" />  <OrderItem Name="草料" Count="2" />  </Item>  </Order> |

读取代码如下：

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Xml.Linq;  namespace \_04\_读取  {  class Program  {  static void Main(string[] args)  {  //加载xml文件  XDocument document = XDocument.Load("Order.xml");  //获取根节点  XElement root = document.Root;  Console.WriteLine(root.Name.ToString());  //获取名为CustomerName的节点，放到IEnumerable泛型中。  IEnumerable<XElement> customer = root.Elements("CustomerName");  //获取名为OrderNumber的节点，放到IEnumerable泛型中。  IEnumerable<XElement> orderNumber = root.Elements("OrderNumber");  //获取名为Item的节点，放到IEnumerable泛型中。  IEnumerable<XElement> item = root.Elements("Item");  //遍历节点  for (int i = 0; i < customer.Count<XElement>(); i++)  {  Console.WriteLine("客户：{0}", customer.ElementAt<XElement>(i).Value);  Console.WriteLine("订单编号：{0}", orderNumber.ElementAt<XElement>(i).Value);  //获取item的子节点  IEnumerable<XElement> orderitem = item.ElementAt<XElement>(i).Elements();  //遍历item的子节点  foreach (XElement element in orderitem)  {  Console.WriteLine("订购商品：{0}\t订购数量：{1}", element.Attribute("Name").Value, element.Attribute("Count").Value);  }  }  Console.ReadLine();  }  }  } |

### 案例：读取流水号

xml文件格式如下：



读取代码如下：

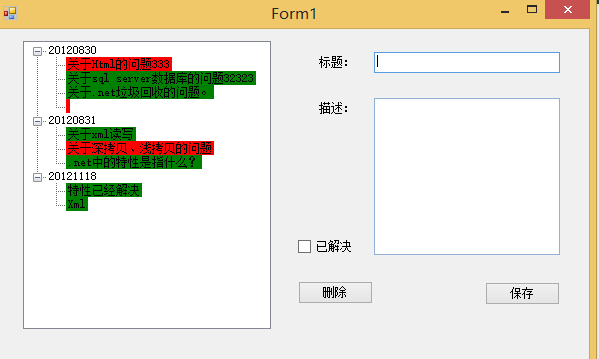
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Xml.Linq;  namespace \_05\_交易流水号读取  {  class Program  {  static void Main(string[] args)  {  //1.读取xml文件  XDocument document = XDocument.Load("ytbank.xml");  //2.读取根节点  XElement root = document.Root;  //3.根据根节点，获得MSG  //4.遍历  foreach (XElement item in root.Elements())  {  foreach (XElement subItem in item.Elements())  {  Console.WriteLine("{0}:\t{1}", subItem.Name, subItem.Attribute("val").Value);  }  Console.WriteLine("=====================================================");  }  Console.ReadLine();  }  }  } |

案例:添加xml各个节点到TreeView中

|  |
| --- |
| private void Form1\_Load(object sender, EventArgs e)  {  XDocument document = XDocument.Load("rss\_sportslq.xml");  XElement root = document.Root;  TreeNode rootNode = treeView1.Nodes.Add(root.Name.ToString());  LoadXmlNode(root.Elements(), rootNode.Nodes);  }  //将xml的节点集合到TreeView节点下。  private void LoadXmlNode(IEnumerable<XElement> elements, TreeNodeCollection treeNodes)  {  //遍历IEnumerable  foreach (XElement element in elements)  { //把节点的名称添加到treeview节点上。  TreeNode node = treeNodes.Add(element.Name.ToString());  //如果没有子元素，把该元素的值添加到treeview中，否则将值添加名称到treeview中。  if (!element.HasElements)  {  node.Nodes.Add(element.Value.ToString());  }  else  {  LoadXmlNode(element.Elements(), node.Nodes);  }  }  } |

### 案例：TreeView中信息的增删改查

界面如下：



增删改查的代码：

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Xml.Linq;  namespace \_08\_知识点记录与修改  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  loadxml();  //默认选择标题文本框  this.ActiveControl = txtTitle;  }  //加载xml到treeview中  private void loadxml()  { //先清理treeview  treeView1.Nodes.Clear();  //加载QQ.xml文件  XDocument document = XDocument.Load("QQ.xml");  XElement root = document.Root;  var dates = root.Elements("Date");  //遍历Date节点把id特性的值添加到treeview上。  foreach (XElement date in dates)  {  TreeNode node = treeView1.Nodes.Add(date.Attribute("id").Value.ToString());  //日期节点标签  node.Tag = 1;  //遍历Q节点  foreach (XElement Q in date.Elements())  {  //把title节点的值添加到treeview上  TreeNode titleNode = node.Nodes.Add(Q.Element("title").Value.ToString());  //设置title节点的标签  titleNode.Tag = 2;  //根据Q的isok属性设置背景色。  bool isok = Boolean.Parse(Q.Attribute("isOk").Value.ToLower());  titleNode.BackColor = isok ? Color.Green : Color.Red;  }  }  //扩展所有节点。  treeView1.ExpandAll();  }  /// <summary>  /// 保存信息到xml文件  /// </summary>  /// <param name="sender"></param>  /// <param name="e"></param>  private void btnSave\_Click(object sender, EventArgs e)  {  //1.先加载xml文件  XDocument document = XDocument.Load("QQ.xml");  XElement root = document.Root;  string title = txtTitle.Text.Trim();  string desc = txtDesc.Text.Trim();  //checkbox 是否已经选择  string isOk = cbIsOk.Checked.ToString();  //是否选择Treeview中的节点  TreeNode node = treeView1.SelectedNode;  //如果没有选择节点  if (node == null)  { //添加一个新的Date节点，属性id的值为  string date = DateTime.Now.ToString("yyyyMMdd");  int count = root.Elements("Date").Where(d => d.Attribute("id").Value.ToString() == date).Count();  //判断节点中是否有该日期，如果有，则添加到该日期节点下。  if (count > 0)  {  IEnumerable<XElement> IQ = root.Elements("Date").Where(d => d.Attribute("id").Value == date);  XElement dateElement = IQ.ElementAt(0);  XElement Q = new XElement("Q");  Q.SetAttributeValue("isOk", isOk);  Q.SetElementValue("title", title);  Q.SetElementValue("desc", desc);  dateElement.Add(Q);  }  else  {//新建一个日期元素，把相关信息添加到其下。  XElement dateElement = new XElement("Date");  dateElement.SetAttributeValue("id", date);  root.Add(dateElement);  XElement Q = new XElement("Q");  Q.SetAttributeValue("isOk", isOk);  Q.SetElementValue("title", title);  Q.SetElementValue("desc", desc);  dateElement.Add(Q);  }  }  else  { //如果选中  int tag = Convert.ToInt32(node.Tag);  //如果选中日期节点，则添加信息到其节点下  if (tag == 1)  {  string date = node.Text;  XElement dateElement = root.Elements("Date").Where(d => d.Attribute("id").Value == date).First();  XElement Q = new XElement("Q");  Q.SetAttributeValue("isOk", isOk);  Q.SetElementValue("title", title);  Q.SetElementValue("desc", desc);  dateElement.Add(Q);  }  else if (tag == 2)  { //选中的是标题则会修改内容  string date = node.Parent.Text;  string oldTitle = node.Text;  XElement dateElement = root.Elements("Date").Where(d => d.Attribute("id").Value == date).First();  XElement titleElement = dateElement.Descendants("title").Where(t => t.Value == oldTitle).First().Parent;  titleElement.SetAttributeValue("isOk", isOk);  titleElement.SetElementValue("title", title);  titleElement.SetElementValue("desc", desc);  }  }  document.Save("QQ.xml");  loadxml();  }  /// <summary>  /// 选择节点，其形象会显示到文本框中  /// </summary>  /// <param name="sender"></param>  /// <param name="e"></param>  private void treeView1\_AfterSelect(object sender, TreeViewEventArgs e)  {  XDocument document = XDocument.Load("QQ.xml");  XElement root = document.Root;  try  {  XElement selectedTitleParent = root.Descendants("title").Where(d => d.Value == e.Node.Text).First().Parent;  txtTitle.Text = e.Node.Text;  txtDesc.Text = selectedTitleParent.Element("desc").Value;  cbIsOk.Checked = bool.Parse(selectedTitleParent.Attribute("isOk").Value);  }  catch (Exception)  {  txtTitle.Text = "";  txtDesc.Text = "";  cbIsOk.Checked = false;  }  }  /// <summary>  /// 选择节点，删除内容  /// </summary>  /// <param name="sender"></param>  /// <param name="e"></param>  private void btnDelete\_Click(object sender, EventArgs e)  {  TreeNode node = treeView1.SelectedNode;  XDocument document = XDocument.Load("QQ.xml");  XElement root = document.Root;  if (node != null)  {  string title = node.Text;  int tag = Convert.ToInt32(node.Tag);  if (tag == 1)  {  XElement dateElement = root.Elements("Date").Where(d => d.Attribute("id").Value == title).First();  dateElement.Remove();  }  else  {  XElement titleElement = root.Descendants("title").Where(t => t.Value == title).First();  titleElement.Parent.Remove();  }  }  else  {  MessageBox.Show("没有选择节点，不能删除");  return;  }  document.Save("QQ.xml");  loadxml();  }  }  } |