**XML**

Создание документов XML осуществляется с помощью:

1. **Sysytem.XML**:

Для работы с XML в C# можно использовать несколько подходов. В первых версиях фреймворка основной функционал работы с XML предоставляло пространство имен System.Xml. В нем определен ряд классов, которые позволяют манипулировать xml-документом:

* **XmlNode**: представляет узел xml. В качестве узла может использоваться весь документ, так и отдельный элемент
* **XmlDocument**: представляет весь xml-документ
* **XmlElement**: представляет отдельный элемент. Наследуется от класса XmlNode
* **XmlAttribute**: представляет атрибут элемента
* **XmlText**: представляет значение элемента в виде текста, то есть тот текст, который находится в элементе между его открывающим и закрывающим тегами
* **XmlComment**: представляет комментарий в xml
* **XmlNodeList**: используется для работы со списком узлов

Ключевым классом, который позволяет манипулировать содержимым xml, является **XmlNode**, поэтому рассмотрим некоторые его основные методы и свойства:

* Свойство **Attributes** возвращает объект XmlAttributeCollection, который представляет коллекцию атрибутов
* Свойство **ChildNodes** возвращает коллекцию дочерних узлов для данного узла
* Свойство **HasChildNodes** возвращает true, если текущий узел имеет дочерние узлы
* Свойство **FirstChild** возвращает первый дочерний узел
* Свойство **LastChild** возвращает последний дочерний узел
* Свойство **InnerText** возвращает текстовое значение узла
* Свойство **InnerXml** возвращает всю внутреннюю разметку xml узла
* Свойство **Name** возвращает название узла. Например, <user> - значение свойства Name равно "user"
* Свойство **ParentNode** возвращает родительский узел у текущего узла

**Редактирование XML**:

Для редактирования xml-документа (изменения, добавления, удаления элементов) мы можем воспользоваться методами класса XmlNode:

* **AppendChild**: добавляет в конец текущего узла новый дочерний узел
* **InsertAfter**: добавляет новый узел после определенного узла
* **InsertBefore**: добавляет новый узел до определенного узла
* **RemoveAll**: удаляет все дочерние узлы текущего узла
* **RemoveChild**: удаляет у текущего узла один дочерний узел и возвращает его

Класс XmlElement, унаследованный от XmlNode, добавляет еще ряд методов, которые позволяют создавать новые узлы:

* **CreateNode**: создает узел любого типа
* **CreateElement**: создает узел типа XmlDocument
* **CreateAttribute**: создает узел типа XmlAttribute
* **CreateTextNode**: создает узел типа XmlTextNode
* **CreateComment**: создает комментарий

**XPath**:

XPath представляет язык запросов в XML. Он позволяет выбирать элементы, соответствующие определенному селектору.

Рассмотрим некоторые наиболее распространенные селекторы:

. выбор текущего узла

.. выбор родительского узла

\* выбор всех дочерних узлов текущего узла

person выбор всех узлов с определенным именем, в данном случае с именем "person"

@name выбор атрибута текущего узла, после знака @ указывается название атрибута (в данном случае "name")

@+ выбор всех атрибутов текущего узла

element[3] выбор определенного дочернего узла по индексу, в данном случае третьего узла

//person выбор в документе всех узлов с именем "person"

person[@name='Иван'] выбор элементов с определенным значением атрибута. В данном случае выбираются все элементы "person" с атрибутом name='Иван'

person[status='женат'] выбор элементов с определенным значением вложенного элемента. В данном случае выбираются все элементы "person", у которых дочерний элемент "status" имеет значение 'женат'

//person/status выбор в документе всех узлов с именем "status", которые находятся в элементах "person"

Действие запросов XPath основано на применении двух методов класса XmlElement:

* **SelectSingleNode**(): выбор единственного узла из выборки. Если выборка по запросу содержит несколько узлов, то выбирается первый
* **SelectNodes**(): выборок по запросу коллекции узлов в виде объекта XmlNodeList

1. **LINQ to XML**:

Еще один подход к работе с Xml представляет технология LINQ to XML. Вся функциональность LINQ to XML содержится в пространстве имен System.Xml.Linq. Рассмотрим основные классы этого пространства имен:

* **XAttribute**: представляет атрибут xml-элемента
* **XComment**: представляет комментарий
* **XDocument**: представляет весь xml-документ
* **XElement**: представляет отдельный xml-элемент

Ключевым классом является XElement, который позволяет получать вложенные элементы и управлять ими. Среди его методов можно отметить следующие:

* **Add**(): добавляет новый атрибут или элемент
* **Attributes**(): возвращает коллекцию атрибутов для данного элемента
* **Elements**(): возвращает все дочерние элементы данного элемента
* **Remove**(): удаляет данный элемент из родительского объекта
* **RemoveAll**(): удаляет все дочерние элементы и атрибуты у данного элемента

Начальный код для XMLFile1:

<?xml version="1.0" encoding="utf-8"?>

<people>

<person name="Игорь">

<age>23</age>

<status>женат</status>

</person>

<person name="Катя">

<age>18</age>

<status>не замужем</status>

</person>

</people>

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

namespace ConsoleApplication34

{

class Person

{

public string Name;

public int Age;

public string Status;

public Person()

{

Name = "Никита";

Age = 10;

Status = "свободен";

}

}

class Phone

{

public string Name { get; set; }

public string Price { get; set; }

}

class Program

{

static void Main(string[] args)

{

//Создание через XMLDoc

#region [Пример 1. Чтение XML-файла]

//XmlDocument xDoc = new XmlDocument(); //создаем объект класса XmlDocument для работы с файлом

//xDoc.Load("XMLFile1.xml");//загружает xml документ из указанного адреса

//XmlElement xRoot = xDoc.DocumentElement;

//foreach (XmlNode xnode in xRoot) // просмотр всех узлов в корневом элементе

//{

// if (xnode.Attributes.Count > 0) // если файл имеет атрибуты

// {

// XmlNode attr = xnode.Attributes.GetNamedItem("name"); //создаем переменную-атрибут name

// if (attr != null) //если атрибут не null

// Console.WriteLine(attr.Value); //выводим значение атрибута

// }

// foreach (XmlNode childnode in xnode.ChildNodes) // для всех остальных элементов

// {

// if (childnode.Name == "age")// если узел - age

// {

// Console.WriteLine("Возраст: {0}", childnode.InnerText);

// }

// if (childnode.Name == "status")// если узел - status

// {

// Console.WriteLine("Статус: {0}", childnode.InnerText);

// }

// }

// Console.WriteLine();

//}

#endregion

#region [Пример 2. Чтение XML-файла и создание объектов]

//List<Person> people = new List<Person>();//лист будующих пиплов

//XmlDocument xDoc = new XmlDocument(); //создаем объект класса XmlDocument для работы с файлом

//xDoc.Load("XMLFile1.xml");//загружает xml документ из указанного адреса

//XmlElement xRoot = xDoc.DocumentElement;

//foreach (XmlNode xnode in xRoot) // просмотр всех узлов в корневом элементе

//{

// Person pers = new Person(); //новый персон

// if (xnode.Attributes.Count > 0) // если файл имеет атрибуты

// {

// XmlNode attr = xnode.Attributes.GetNamedItem("name"); //создаем переменную-атрибут name

// if (attr != null) //если атрибут не null

// pers.Name = attr.Value;

// Console.WriteLine(attr.Value); //выводим значение атрибута

// }

// foreach (XmlNode childnode in xnode.ChildNodes) // для всех остальных элементов

// {

// if (childnode.Name == "age")// если узел - age

// {

// pers.Age = Convert.ToInt32(childnode.InnerText);

// Console.WriteLine("Возраст: {0}", childnode.InnerText);

// }

// if (childnode.Name == "status")// если узел - status

// {

// pers.Status = childnode.InnerText;

// Console.WriteLine("Статус: {0}", childnode.InnerText);

// }

// }

// people.Add(pers);

// Console.WriteLine();

//}

#endregion

#region [Пример 3. Добавление объектов в XML]

//XmlDocument xDoc = new XmlDocument(); //создаем объект класса XmlDocument для работы с файлом

//xDoc.Load("XMLFile1.xml");//загружает xml документ из указанного адреса

//XmlElement xRoot = xDoc.DocumentElement;

//XmlElement personElem = xDoc.CreateElement("person");// создаем новый элемент person

//XmlAttribute nameAttr = xDoc.CreateAttribute("name");// создаем атрибут name

//XmlElement ageElem = xDoc.CreateElement("age");// создаем элемент age

//XmlElement statusElem = xDoc.CreateElement("status");// создаем элемент status

//// создаем текстовые значения для элементов и атрибута

//XmlText nameText = xDoc.CreateTextNode("Зина");

//XmlText ageText = xDoc.CreateTextNode("30");

//XmlText statusText = xDoc.CreateTextNode("в разводе");

////добавляем узлы

//nameAttr.AppendChild(nameText);

//statusElem.AppendChild(statusText);

//ageElem.AppendChild(ageText);

//personElem.Attributes.Append(nameAttr);

//personElem.AppendChild(statusElem);

//personElem.AppendChild(ageElem);

//xRoot.AppendChild(personElem);

//xDoc.Save("XMLFile1.xml");

//Console.WriteLine("Все четко");

#endregion

#region [Пример 4. Удаление объектов из XML]

//XmlDocument xDoc = new XmlDocument();

//xDoc.Load("XMLFile1.xml");

//XmlElement xRoot = xDoc.DocumentElement;

//XmlNode firstNode = xRoot.FirstChild;

//xRoot.RemoveChild(firstNode);

//xDoc.Save("XMLFile1.xml");

//Console.WriteLine("Удолили");

#endregion

#region [Пример 5. XPath]

//XmlDocument xDoc = new XmlDocument();

//xDoc.Load("XMLFile1.xml");

//XmlElement xRoot = xDoc.DocumentElement;

//XmlNodeList childnodes = xRoot.SelectNodes("\*");// выбор всех дочерних узлов

//foreach (XmlNode n in childnodes)

// Console.WriteLine(n.OuterXml);

//XmlNodeList childnodes1 = xRoot.SelectNodes("person");//вывод значений атрибутов name у персонов

//foreach (XmlNode n in childnodes1)

// Console.WriteLine(n.SelectSingleNode("@name").Value);

//XmlNode childnode = xRoot.SelectSingleNode("person[@name='Катя']");//выбор узла, где имя Катя

//if (childnode != null)

// Console.WriteLine(childnode.OuterXml);

#endregion

//Создание через LINQ to XML

#region [Пример 6. Создание документа XML]

//XDocument xdoc = new XDocument();

//// создаем первый элемент

//XElement iphone6 = new XElement("phone");

//// создаем атрибут

//XAttribute iphoneNameAttr = new XAttribute("name", "iPhone 6");

//XElement iphoneCompanyElem = new XElement("company", "Apple");

//XElement iphonePriceElem = new XElement("price", "40000");

//// добавляем атрибут и элементы в первый элемент

//iphone6.Add(iphoneNameAttr);

//iphone6.Add(iphoneCompanyElem);

//iphone6.Add(iphonePriceElem);

//// создаем второй элемент

//XElement galaxys5 = new XElement("phone");

//XAttribute galaxysNameAttr = new XAttribute("name", "Samsung Galaxy S5");

//XElement galaxysCompanyElem = new XElement("company", "Samsung");

//XElement galaxysPriceElem = new XElement("price", "33000");

//galaxys5.Add(galaxysNameAttr);

//galaxys5.Add(galaxysCompanyElem);

//galaxys5.Add(galaxysPriceElem);

//// создаем корневой элемент

//XElement phones = new XElement("phones");

//// добавляем в корневой элемент

//phones.Add(iphone6);

//phones.Add(galaxys5);

//// добавляем корневой элемент в документ

//xdoc.Add(phones);

////сохраняем документ

//xdoc.Save("phones.xml");

////Можно быстрее

//XDocument xdoc = new XDocument(new XElement("phones",

//new XElement("phone",

//new XAttribute("name", "iPhone 6"),

//new XElement("company", "Apple"),

//new XElement("price", "40000")),

//new XElement("phone",

//new XAttribute("name", "Samsung Galaxy S5"),

//new XElement("company", "Samsung"),

//new XElement("price", "33000"))));

//xdoc.Save("phones.xml");

#endregion

#region [Пример 7. Чтение и выборка через LINQ]

////Чтение

//XDocument xdoc = XDocument.Load("phones.xml");

//foreach (XElement phoneElement in xdoc.Element("phones").Elements("phone"))

//{

// XAttribute nameAttribute = phoneElement.Attribute("name");

// XElement companyElement = phoneElement.Element("company");

// XElement priceElement = phoneElement.Element("price");

// if (nameAttribute != null && companyElement != null && priceElement != null)

// {

// Console.WriteLine("Смартфон: {0}", nameAttribute.Value);

// Console.WriteLine("Компания: {0}", companyElement.Value);

// Console.WriteLine("Цена: {0}", priceElement.Value);

// }

// Console.WriteLine();

//}

////Выборка всех, кто от компании Samsung

//var items = from xe in xdoc.Element("phones").Elements("phone")

// where xe.Element("company").Value == "Samsung"

// select new Phone

// {

// Name = xe.Attribute("name").Value,

// Price = xe.Element("price").Value

// };

//foreach (var item in items)

// Console.WriteLine("{0} - {1}", item.Name, item.Price);

#endregion

#region [Пример 8. Изменение файла через LINQ]

//XDocument xdoc = XDocument.Load("phones.xml");

//XElement root = xdoc.Element("phones");

//foreach (XElement xe in root.Elements("phone").ToList())

//{

// // изменяем название и цену

// if (xe.Attribute("name").Value == "Samsung Galaxy S5")

// {

// xe.Attribute("name").Value = "Samsung Galaxy Note 4";

// xe.Element("price").Value = "31000";

// }

// //если iphone - удаляем его

// else if (xe.Attribute("name").Value == "iPhone 6")

// {

// xe.Remove();

// }

//}

//// добавляем новый элемент

//root.Add(new XElement("phone",

// new XAttribute("name", "Nokia Lumia 930"),

// new XElement("company", "Nokia"),

// new XElement("price", "19500")));

//xdoc.Save("pnones1.xml");

//// выводим xml-документ на консоль

//Console.WriteLine(xdoc);

#endregion

Console.ReadKey();

}

}

}