Лабораторная работа №8

Обработка XML документов. Компоненты отображения списков и деревьев.

1) XML документ

Язык разметки документов – это набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры. Тэги языка, или, как их иногда называют, управляющие дескрипторы, в таких документах каким-то образом кодируются, выделяются относительно основного содержимого документа и служат в качестве инструкций для программы, производящей показ содержимого документа на стороне клиента. В самых первых системах для обозначения этих команд использовались символы “<” и “>”, внутри которых помещались названия инструкций и их параметры. Сейчас такой способ обозначения тэгов является стандартным.

Пример простого XML документа:

```
<?xml version='1.0' encoding='windows-1251'?>
<group>
   <student>Иванов Иван</student>
   <student>Петров Александр</student>
   <student>Сидоров Евгений</student>
</group>
```

Строка <?xml version='1.0' encoding='windows-1251'?> называется объявлением документа. В этом документе имеются теги двух видов – <group> и <student>. Тэг вида <имя_тэга> (открывающий) является началом элемента с именем <имя_тэга>, а тэг вида </имя тэга> (закрывающий) закрывает элемент. Элемент group является корневым для этого документа. Наличие одного (и только одного) корневого элемента является обязательным правилом XML.

Информация, которую хранит элемент, помещают между открывающим и закрывающим тэгами.

При назначении имен следует придерживаться следующих правил:

1) имя элемента должно начинаться с буквы или символа подчеркивания,

после чего могут идти буквы, цифры, символы точки, тире или подчеркивания;

2) имена, начинающиеся с префикса xml, зарезервированы для стандартных имен;

3) двоеточие в имени зарезервировано для задания пространства имен.

Если при определении элементов необходимо задать какие-либо параметры, уточняющие его характеристики, то имеется возможность использовать атрибуты элемента. Атрибут – это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге. Пример:

```
<?xml version='1.0' encoding='windows-1251'?>
<group>
   <student name=”Иван” age=”21”>Иванов</student>
   <student name=”Александр” age=”20”>Петров</student>
   <student name=”Евгений” age=”21”>Сидоров</student>
</group>
```

XML документы часто используются для передачи информации в сети Интернет. Например, главные новости газеты «Ведомости» (http://www.vedomosti.ru/) могут быть получены из XML документа http://www.vedomosti.ru/newspaper/out/rss.xml.

2) Средства языка C# для работы с XML документами

Для работы с XML документами в языке C# в пространстве имен System.Xml объявлен ряд классов. Наиболее удобным является класс XElement (http://msdn.microsoft.com/ru-ru/library/system.xml.linq.xelement(v=vs.110).aspx), так как он поддерживает загрузку данных из сети Интернет и Linq запросы. Для обработки документа достаточно вызвать статический метод Load:

```
XElement rootTag =
            XElement.Load("http://www.vedomosti.ru/newspaper/out/rss.xml");
```

Основные члены класса XElement:

методы:

1) Elements, возвращает все дочерние элементы,

2) Element, возвращает дочерний элемент с заданным именем Attributes, возвращает список атрибутов

3) Attribute, возвращает атрибут с заданным именем

свойства:

1) Name (типа XName) – имя элемента;

2) Parent (типа XElement) – родительский элемент;

3) Value (типа string) – текстовое содержимое элемента;

Класс XName используется для представления имен тэгов и атрибутов. Для создания элемента XName из объекта класса string используется статический метод Get.

3) Компоненты отображения списков и деревьев

Для отображения деревьев в языке C# имеется компонент TreeView (http://msdn.microsoft.com/ru-ru/library/system.windows.forms.treeview(v=vs.110).aspx), который отображает иерархию объектов TreeNode. Для задания объектов для отображения их достаточно добавить в коллекцию Nodes компонента TreeView.

Пример кода для отображения простейшего дерева в дереве tvTree:

```
public partial class frmMain : Form
{
    public frmMain()
    {
        InitializeComponent();

        TreeNode rootNode = new TreeNode("ROOT"); //Создание корневого элемента
        for (int i = 1; i <= 5; i++) //Цикл для дочерних элементов
            rootNode.Nodes.Add(String.Format("Child #{0}", i));
        tvTree.Nodes.Add(rootNode); //Добавление корневого элемента в дерево
    }
}
```

Для отображения списков используется компонент ListView (http://msdn.microsoft.com/ru-ru/library/system.windows.forms.listview(v=vs.90).aspx). Данные для отображения задаются в коллекции Items, элементы коллекции имеют тип ListViewItem. Компонент ListView может работать в нескольких режимах, широко известных по работе с Проводником ОС Windows:

1) крупные значки

2) мелкие значки

3) список

4) таблица

Режим работы выбирается с помощью свойства View.

Ниже приведен пример программы, осуществляющей синтаксический анализ XML документа, и отображающий результат в виде дерева тэгов и таблицы с некоторыми свойствами тэгов:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;

namespace Linq
{
    public partial class frmMain : Form
    {
        public frmMain()
        {
            InitializeComponent();

            //Загрузка ленты
            XElement rootTag =
            XElement.Load("http://www.vedomosti.ru/newspaper/out/rss.xml");
            //Создание корневого элемента
            TreeNode rootNode = new TreeNode(rootTag.Name.ToString());
            //Поиск дочерних элементов корневого элемента
            FindChildrenTags(rootNode, rootTag);
            //Добавление корневого элемента в дерево
            tvTree.Nodes.Add(rootNode);
        }

        /// <summary>
        /// Поиск и добавление дочерних элементов
        /// </summary>
        /// <param name="node">Корневой узел дерева</param>
        /// <param name="tag">Корневой тэг</param>
        private void FindChildrenTags(TreeNode node, XElement tag)
        {
            node.Tag = tag; //Связь Узла и Тэга

            //Получение дочерних элементов
            IEnumerable<XElement> childTags = tag.Elements();
            if (childTags != null) //Если дочерние элементы есть
            {
                foreach (XElement childTag in childTags)
                {
                    //Создание дочернего узла
                    TreeNode childNode = new TreeNode(childTag.Name.ToString());
                    //Побавление поддочерних элементов
                    FindChildrenTags(childNode, childTag);
                    //Добавление дочернего узла в дерево
```

```
                node.Nodes.Add(childNode);
            }
        }
    }

    //Вызывается при щелчке по узлу в дереве
    private void tvTree_AfterSelect(object sender, TreeViewEventArgs e)
    {
        //Получение объета, связанно с узлом
        object tag = e.Node.Tag;
        //Проверка на принадлежность к классу XElement
        if (tag is XElement)
        {
            //Приведение типа
            XElement element = (XElement)tag;
            //Если элемент имеет имя "item"
            if ("item".Equals(element.Name.ToString().ToLower()))
            {
                //Добавление названия столбцов
                lvInfo.Items.Clear();
                lvInfo.Columns.Clear();

                lvInfo.Columns.Add("Название", 200);
                lvInfo.Columns.Add("Дата", 100);
                lvInfo.Columns.Add("Ссылка", 200);

                //Заполение таблицы
                XElement title = element.Element(XName.Get("title"));
                ListViewItem item = new ListViewItem(title.Value);

                XElement date = element.Element(XName.Get("pubDate"));
                item.SubItems.Add(date.Value);

                XElement link = element.Element(XName.Get("link"));
                item.SubItems.Add(link.Value);

                lvInfo.Items.Add(item);
            }
        }
    }
}
```

Результат работы программы приведен на рисунке 1.

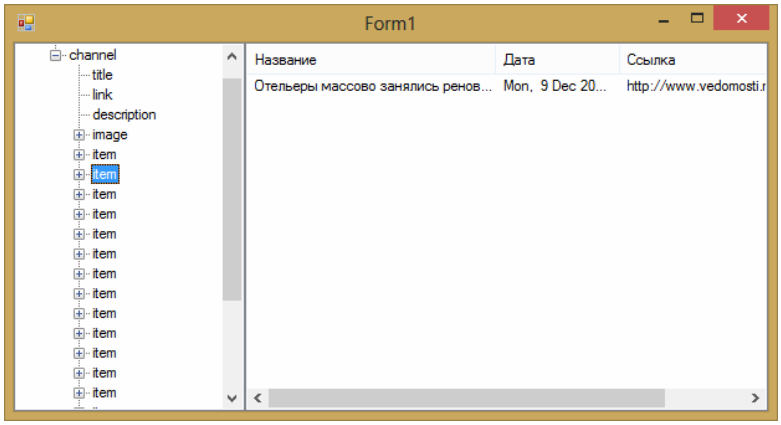

Рисунок 1 – Пример работы программы синтаксического разбора XML документа

4) Выполнение лабораторной работы №8

1) Внимательно изучите XML документ, согласно Вашему варианту заданий.

2) Напишите простейшую программу, которая открывает XML документ с помощью конструктора XElement:

XElement rootTag = XElement.Load(ваша_ссылка);

3) Запустите ее в режиме отладки и изучите объект rootTag.

4) Напишите программу, которая в удобном виде выводит содержимое rss ленты согласно Вашему варианту задания. **TreeView и ListView в работе НЕ ИСПОЛЬЗОВАТЬ.** Приветствуется наличие интуитивно понятного интерфейса и представление большого объема информации (показ содержимого максимально возможного количества тэгов). Загрузка ленты должна производиться в фоновом потоке.

5) Задания к лабораторной работе №8

1) Написать программу синтаксического разбора rss-ленты «Пятница» газеты «Ведомости»: http://www.vedomosti.ru/out/friday.xml

2) Написать программу синтаксического разбора rss-ленты «Горячие документы» системы «КонсультантПлюс»: http://www.consultant.ru/rss/hotdocs.xml

3) Написать программу синтаксического разбора rss-ленты «Яндекс.Новости: В мире»: http://news.yandex.ru/world.rss

4) Написать программу синтаксического разбора rss-ленты «Анекдоты про Вовочку»: http://www.anekdot.ru/rss/tag/2.xml

4) Написать программу синтаксического разбора rss-ленты «Анекдоты про Чапаева»: http://www.anekdot.ru/rss/tag/1.xml

5) Написать программу синтаксического разбора rss-ленты «Lenta.ru: Статьи»: http://lenta.ru/rss/articles/

6) Написать программу синтаксического разбора rss-ленты «Мир Формулы-1»: http://www.f1-world.ru/news/rssexp6.xml

7*) Разработать программу для получения информации о погоде из формата XML (документация и вся информация расположена на сайте https://www.meteoservice.ru/content/export). В программе должно быть реализовано следующее:

а) пользователь может выбрать в интерфейсе один из трёх городов: Волгоград, Москва и любой другой третий; в зависимости от этого программа получает с сайта необходимый XML-документ;

б) в программе реализована структура данных (класс) для описания одного прогноза, соответствующая структуре элемента FORECAST из XML файла;

в) в программе должна быть реализована функция, для считывания из XML-файла всех имеющихся в нём прогнозов в виде списка элементов класса "прогноз";

г) прогноз погоды выводится на экран в Нормальном, Удобном для Людей (не технарей), Принятом для подобного рода информации виде (по аналогии с тем как это выдаёт Google по запросу «прогноз погоды», gismeteo и т. д.);

д) прогноз выводится на экран сразу после запуска программы не дожидаясь нажатия на ненужные непонятные кнопки, вроде "отправить запрос". Затем уже пользователь, при необходимости, может выбрать другой город.

8**) Считать автоматически цвета с изображения палитры (https://pxc-spb.ru/img_catalog/komlekt-krovla/dobor/RAL_2.gif) и представить эту палитру в формате XML-документа. Потом этот документ необходимо считать и вывести палитру на его основе. Помимо предложенной, можно использовать другие палитры с сайта http://www.pxc-spb.ru/img_catalog/komlekt-krovla/dobor/ral.html.

6) Контрольные вопросы

1) Что такое XML и для чего он применяется?
2) Какова структура XML-документа?
3) Какие средства для работы с XML-документами имеются в C#
4) Опишите класс XElement.
5) Опишите элементы управления для отображения списков и деревьев.