Язык XML и объектные модели XML-документа

Язык XML

Язык XML (eXtended Markup Language) — универсальный язык разметки структурированных (иерархических) данных. «XML — это платформенно-, программно- и аппаратно-независимое средство для передачи информации».

Пример XML-документа.

```
<?xml version="1.0" encoding="windows-1251"?>
<описания-книг>
<!-- Использованы теги FB2 (Fiction Book 2.0) -->
  <title-info>
    <genre match="90">sf_fantasy</genre>
    <author>
      <first-name>Джон</first-name>
      <middle-name>Рональд Руэл</middle-name>
      <last-name>Толкин</last-name>
    </author>
    <book-title>Возвращение Короля</book-title>
    <lang>ru</lang>
    <src-lang>en</src-lang>
    <translator>
      <first-name>Мария</first-name>
      <last-name>Каменкович</last-name>
    </translator>
    <translator>
      <first-name>Валерий</first-name>
      <last-name>Каррик</last-name>
    </translator>
    <sequence name="Властелин Колец" number="3" />
  </title-info>
  <title-info>
    <genre>sf</genre>
    <author>
      <first-name>Аркадий</first-name>
      <last-name>Стругацкий</last-name>
    </author>
    <author>
      <first-name>Борис</first-name>
      <last-name>Стругацкий</last-name>
    </author>
    <book-title>Улитка на склоне</book-title>
    <lang>ru</lang>
  </title-info>
</описания-книг>
```

Составляющие XML-документа

Как правило, любой XML-документ начинается с *объявления XML* (XML declaration), в котором, в частности, указывается версия стандарта XML и кодировка документа. Объявление XML входит в *пролог XML* (XML prolog), в котором может содержаться дополнительная информация, связанная с документом в целом, например, *объявление типа документа* (type document declaration; в приведенном примере оно отсутствует).

Документ XML должен содержать *корневой элемент* (root element); в приведенном примере это элемент `описания-книг`.

Любой элемент, в том числе корневой, может содержать *дочерние элементы* (child elements), а также *дочерние узлы* других видов (child nodes): например, элемент `описания-книг` содержит два дочерних элемента `title-info`.

Примером содержимого элемента XML, отличного от его дочерних элементов, является *обычный текст*.

Если элемент имеет непустое содержимое, то оно должно располагаться между открывающим и закрывающим *тегом* (tag), например, `<author>` и `</author>`. Если элемент не имеет содержимого, то его можно указывать с помощью комбинированного тега (см. элемент `sequence`).

Элементы могут включать *атрибуты* (attributes) которые указываются сразу за именем элемента в открывающем или комбинированном теге в формате `имя="значение"`.

В любом месте XML-документа можно помещать *комментарий* (comment) — текст, заключенный между тегами `<!--` и `-->`, — и *инструкции обработки* (processing instructions), заключенные между тегами <? и ?>. XML-элементы могут также содержать «буквальные» *символьные данные* CDATA (описываются далее).

Корректные и действительные XML-документы

Любой XML-документ должен удовлетворять набору правил, при нарушении которых он считается *некорректным* (*неправильно сформированным)*. Основные из этих правил:

- если в документе присутствует описание, то оно должно располагаться в начале документа;
- в любом документе должен быть *единственный* корневой элемент;
- помимо буквенных и цифровых символов имя элемента (как и имя атрибута) может содержать только символы «.» (точка), «_» (подчеркивание) и «-» (дефис); прочие символы, в том числе пробелы, не допускаются;
- имена элементов и атрибутов не могут начинаться с цифры, точки или дефиса (но могут начинаться с символа подчеркивания);
- имена элементов и атрибутов чувствительны к регистру;
- любой открывающий тег должен иметь парный ему закрывающий тег; при этом должна обеспечиваться правильная вложенность тегов (например, фрагмент `<author><Author></author></Author>` считается ошибочным); на комбинированный тег вида `<имя />` данное правило не распространяется;
- между начальными символами тегов («<» для открывающего или комбинированного, «</» для закрывающего) и именем элемента не должно быть пробелов (перед символом «>» пробелы допускаются);
- один элемент не может иметь несколько атрибутов с одинаковыми именами (хотя может содержать несколько одноименных дочерних элементов);
- значения любых атрибутов должны заключаться в однотипные кавычки, при этом само значение атрибута не должно содержать кавычки данного типа;
- в числовых значениях в качестве разделителя всегда используется точка;
- для некоторых символов требуется использовать специальные обозначения (*предопределенные символьные сущности*): `&amp;` для «&», `&lt;` для «<», `&gt;` для «>».

Есть еще две предопределенные символьные сущности (достаточно редко используемые): `&apos;` для «'», `&quot;` для «"».

Кроме предопределенных символьных сущностей в XML можно использовать *нумерованные символьные сущности*, которые позволяют указать любой Unicode-символ с помощью его номера. Форматы нумерованных символьных сущностей: `&#`*десятичный_номер*`;` и `&#x`*шестнадцатеричный_номер*`;`. Например, символ «&» (код 38) можно указать тремя способами: `&amp;`, `&#38;` или `&#x26;`.

Помимо основных правил, выполнение которых необходимо для того, чтобы любой XML-документ считался корректным (well-formed), на формат XML-документа могут накладываться дополнительные правила, вытекающие из предметной области, связанной с данным форматом.

Например, можно потребовать, чтобы любой документ, описывающий литературные произведения (подобный приведенному выше), удовлетворял дополнительным условиям:

- любой элемент title-info должен содержать не менее одного элемента author и genre, ровно один элемент book-title и lang, не более одного элемента src-lang и sequence, любое количество элементов translator;
- элемент author должен содержать ровно один элемент last-name и не более одного элемента first-name и middle-name;
- элементы book-title и lang должны иметь только текстовое содержимое, причем содержимое book-title может быть произвольным текстом, а содержимое lang должно совпадать с одним из стандартных имен, определенных для различных языков («ru» для русского, «en» для английского и т. д.);
- элемент genre может содержать дополнительный атрибут weight, значение которого должно представлять собой целое число из диапазона 1–100;
- и так далее.

Все подобные дополнительные правила описываются на специальном языке и могут оформляться либо в виде *определения типа документа* (document type definition, DTD), либо в виде *схемы* (schema) XML-документа. Если XML-документ удовлетворяет требуемому DTD или схеме, то он считается не только корректным, но и *действительным*, или *валидным* (valid).

Объектные модели XML-документа

Все конструкции, входящие в правильно сформированный XML-документ, могут быть представлены классами, образующими *объектную модель документа* (Document Object Model, DOM). Можно говорить также о программном интерфейсе DOM — DOM API. Стандартной моделью DOM является модель W3C DOM, предложенная консорциумом W3C — официальным разработчиком языка XML. Пример (в начале программы надо указать директиву using System.Xml):

```
XmlDocument d = new XmlDocument();
d.AppendChild(d.CreateXmlDeclaration("1.0", "windows-1251", null));
XmlElement bookDescriptions = d.CreateElement("описания-книг");
d.AppendChild(bookDescriptions);
XmlElement titleInfo = d.CreateElement("title-info");
XmlElement genre = d.CreateElement("genre");
XmlAttribute attr = d.CreateAttribute("match");
attr.InnerText = "90";
genre.Attributes.Append(attr);
genre.InnerText = "sf_fantasy";
titleInfo.AppendChild(genre);
XmlElement child = d.CreateElement("author");
XmlElement name = d.CreateElement("first-name");
name.InnerText = "Джон";
child.AppendChild(name);
name = d.CreateElement("middle-name");
name.InnerText = "Рональд Руэл";
child.AppendChild(name);
name = d.CreateElement("last-name");
name.InnerText = "Толкин";
child.AppendChild(name);
titleInfo.AppendChild(child);
child = d.CreateElement("book-title");
child.InnerText = "Возвращение Короля";
titleInfo.AppendChild(child);
child = d.CreateElement("lang");
child.InnerText = "ru";
titleInfo.AppendChild(child);
child = d.CreateElement("sequence");
attr = d.CreateAttribute("name");
attr.InnerText = "Властелин Колец";
child.Attributes.Append(attr);
attr = d.CreateAttribute("number");
attr.InnerText = "3";
child.Attributes.Append(attr);
titleInfo.AppendChild(child);
bookDescriptions.AppendChild(titleInfo);
d.Save("MyBooks.xml");
```

Результат будет записан в файл MyBooks.xml:

```
<?xml version="1.0" encoding="windows-1251"?>
<описания-книг>
  <title-info>
    <genre match="90">sf_fantasy</genre>
    <author>
      <first-name>Джон</first-name>
      <middle-name>Рональд Руэл</middle-name>
      <last-name>Толкин</last-name>
    </author>
    <book-title>Возвращение Короля</book-title>
    <lang>ru</lang>
    <sequence name="Властелин Колец" number="3" />
  </title-info>
</описания-книг>
```

По коду программы сложно определить вид формируемого XML-документа. Обработка полученного документа непосредственно в программе также является непростой задачей. Например, вывести содержимое документа или какого-либо его узла, указав связанный с ним объект в методе WriteLine, не удастся: при выполнении следующего фрагмента

```
Console.WriteLine(d);
Console.WriteLine(titleInfo);
```

будет выведен текст

```
System.Xml.XmlDocument
System.Xml.XmlElement
```

Если исправить фрагмент, обратившись к свойству OuterXml,

```
Console.WriteLine(d.OuterXml);
Console.WriteLine(titleInfo.OuterXml);
```

то в консольном окне будет выведен соответствующий текст *без форматирования*, например:

```
<?xml version="1.0" encoding="windows-1251"?><описания-книг><title-info><genre match="90">sf_fantasy</genre><author><first-name>Джон</first-name><middle-name>Рональд Руэл</middle-name><last-name>Толкин</last-name></author><book-title>Возвращение Короля</book-title><lang>ru</lang><sequence name="Властелин Колец" number="3" /></title-info></описания-книг>
```

Учитывая сложность работы с W3C DOM API, разработчики версии .NET Framework 3.5 включили в нее альтернативную объектную модель для XML-документов — X-DOM. Эта модель, наряду с операторами запросов, оформленными в виде методов расширения класса Extensions из пространства имен System.Xml.Linq, образует интерфейс **LINQ to XML**.

Приведенный ниже фрагмент, формирует тот же документ с использованием X-DOM API:

```
XDocument d = new XDocument(
  new XDeclaration("1.0", "windows-1251", null),
  new XElement("описания-книг",
    new XElement("title-info",
      new XElement("genre",
        new XAttribute("match", "90"),
        "sf_fantasy"),
      new XElement("author",
        new XElement("first-name", "Джон"),
        new XElement("middle-name", "Рональд Руэл"),
        new XElement("last-name", "Толкин")),
      new XElement("book-title", "Возвращение Короля"),
      new XElement("lang", "ru"),
      new XElement("sequence",
        new XAttribute("name", "Властелин Колец"),
        new XAttribute("number", "3")))));
```

Для вывода полученного документа на экран *в отформатированном виде* достаточно выполнить оператор

```
Console.WriteLine(d);
```