ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Информационные технологии»

Учебно-методическое пособие

по дисциплине

«Информационные технологии в отрасли»

«Основы XML»

Автор:

Захарова О.А.

Ростов-на-Дону, 2023

Аннотация

Пособие предназначено для студентов очной формы обучения направления 09.04.02 – Информационные системы и технологии.

Авторы

д.п.н., доцент
Захарова О.А.

Оглавление

ВВЕДЕНИЕ

XML (англ. *e**X**tensible **M**arkup **L**anguage* — расширяемый язык разметки; произносится [*икс-эм-эль*]) — рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для **создания** и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

XML является подмножеством SGML.

1. ЯЗЫК XML

Спецификация XML описывает язык и ряд вопросов, касающихся кодировки и обработки документов. Материал этой секции представляет собой сокращённое изложение описания языка в Спецификации XML, адаптированное для настоящей статьи.

Нормативным считается английский вариант документа, поэтому основные термины приводятся с их английскими оригиналами.

Перевод основных терминов в основном следует доступному в интернете переводу Спецификации на русский язык, исключение составляют термины *tag* и *declaration*.

Для термина tag здесь используется перевод *тег* как соответствующий текущим нормам русского языка.

Для термина declaration отдано предпочтение распространённому переводу *объявление* (против также распространённой кальки *декларация*).

В литературе и интернете могут встречаться и иные переводы основных терминов.

1.1 Физическая и логическая структуры документа

С физической точки зрения документ состоит из сущностей (англ. entities), из которых каждая может отсылать на другую сущность. Единственный **корневой элемент** — документная сущность. Содержание сущностей — символы.

С логической точки зрения документ состоит из комментариев (англ. comments), объявлений (англ. declarations), элементов (англ. elements), ссылок на сущности (англ. character references) и инструкций обработки (англ. processing instructions). Всё это в документе структуризуется разметкой (англ. markup).

1.1.1 Физическая структура

Сущность — мельчайшая часть в документе. Все сущности что-нибудь содержат, и у всех них есть имя (существуют исключения, напр. **документная сущность**). Проще говоря, термин «сущность» описывает «сущую вещь», «**что-то**».

Документ состоит из сущностей, содержание которых — символы. Все они разделены на два типа: *символьных данных* (*англ. character data*) и разметки. К разметке принадлежат: *теги* (*англ. tags*), обозначающие границы элементов, объявления и инструкции обработки, включая их *атрибуты* (*англ. attributes*), ссылки на сущности, комментарии, а также последовательности сим-

волов, обрамляющие секции «CDATA». Часть документа, не принадлежащая разметке, составляет символьные данные документа.

1.1.2 Логическая структура

Все составляющие части документа обобщаются в *пролог* и **корневой элемент**. **Корневой элемент** — обязательная часть документа, составляющая всю его суть (пролог, вообще говоря, может отсутствовать). Может включать (а может не включать) вложенные в него элементы и символьные данные, а также комментарии. Вложенные в корневой элемент элементы, в свою очередь, могут включать вложенные в них элементы, символьные данные и комментарии, и так далее. **Пролог** может включать **объявления**, **инструкции обработки**, **комментарии**. Его следует начинать с *объявления XML*, хотя в определённой ситуации допускается отсутствие этого объявления.

Элементы документа должны быть *правильно вложены*: любой элемент, начинающийся внутри другого элемента (то есть любой элемент документа, кроме корневого), должен заканчиваться внутри элемента, в котором он начался. Символьные данные могут встречаться внутри элементов как непосредственно так и в специальных *секциях «CDATA»*. Объявления, инструкции обработки и элементы могут иметь связанные с ними атрибуты. Атрибуты используются для связывания с логической единицей текста пар имя-значение.

1.2 Символы разметки

Разметка всегда начинается символом < и заканчивается символом >. Наряду с символами < и >, специальную роль для разметки играет также символ &. Угловые скобки обозначают границы элементов, инструкций обработки и некоторых других последовательностей. Амперсанд позволяет выполнить замену текста при помощи *сущностей* (*англ. entities*).

1.3 Решение проблемы неоднозначности разметки

Употребление разметочных символов в символьных данных затрудняет распознавание конструкций разметки и может создать проблему неоднозначности структуры. В XML эта проблема решается следующим образом: три упомянутых символа не могут присутствовать в символьных данных и в значениях атрибутов в их непосредственном виде, для их представления в этих случаях зарезервированы специальные *сущности*:

Символ	Замена
<	<
>	>
&	&

Кроме того, для употребления апострофов и кавычек внутри значений атрибутов используются следующие *сущности*:

'	'
"	"

Правило замены символов, используемых в разметке, на ими обозначаемых *сущностей* не распространяется на символьные данные в секциях «CDATA», зато выполняется во **всех** остальных местах документа.

1.4 Имена

В языке XML все имена должны начинаться с буквы, символа нижнего подчеркивания (_) или двоеточия (:) и продолжаться только допустимыми для имен символами, а именно они могут содержать только буквы, входящие в секцию букв кодировки Unicode, арабские цифры, дефисы, знаки подчеркивания, точки и двоеточия. Однако имена не могут начинаться со строки xml в любом регистре. Имена, начинающиеся с этих символов, зарезервированы для использования консорциумом W3C. Нужно помнить что так как буквы не ограничены исключительно символами ASCII, то в именах можно использовать слова из родного языка.

1.5 Пролог

1.5.1 Объявление XML

Объявление XML указывает версию языка, на которой написан документ. Поскольку интерпретация содержимого документа, вообще говоря, зависит от версии языка, то Спецификация предписывает начинать документ с объявления XML. В первой (1.0) версии языка использование объявления не было обязательным, в последующих версиях оно обязательно. Таким образом, версия языка определяется из объявления, и если объявление отсутствует, то принимается версия 1.0.

Кроме версии XML, объявление может также содержать информацию о *кодировке* документа и «оставаться ли документу со своим собственным DTD, или с подключённым».

Пример:

```
<?xml version="1.1" encoding="UTF-8" ?>
```

или:

<?xml version="1.0" encoding="windows-1251"?>

Во всех этих примерах отсутствовал атрибут «standalone», который как раз и определяет, подключить ли документу описания разметки извне. По умолчанию он равен «no»:

<?xml version="1.0" encoding="windows-1251" standalone="yes"?>

Это значит, документ будет пользоваться своим DTD.

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

Так мы явно задаём, что DTD подключается извне.

1.5.2 Объявление типа документа

Для объявления типа документа существует специальная инструкция !DOCTYPE. Она позволяет задать при помощи языка DTD, какие в документ входят элементы, каковы их атрибуты, какие сущности могут использоваться и кое-что ещё.

Например, вот корректный документ:

<?xml version="1.0"?>

<greeting>Hello, world!</greeting>

В нём есть корневой элемент <greeting>Hello, world!</greeting>, и с логической точки зрения документ существует. Однако он недействителен (*англ. not valid*).

При помощи Объявления типа документа (DTD) возможно описывать его содержания и логическую структуру, а также связывать с определённым элементом пару «имя — значение».

Запишем, как выглядит пролог, используя запись Бэкуса — Наура:

```
prolog        ::= XMLDecl? Misc* (doctypedecl Misc*)?
XMLDecl       ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
VersionInfo   ::= S 'version' Eq ("'" VersionNum "'" | '"' VersionNum '"')
Eq            ::= S? '=' S?
VersionNum    ::= '1.' [0-9]+
Misc          ::= Comment | PI | S
doctypedecl   ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('[' intSubset ']' S?)? '>'
DeclSep       ::= PEReference | S
intSubset     ::= (markupdecl | DeclSep)*
markupdecl    ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment
extSubset     ::= TextDecl? extSubsetDecl
extSubsetDecl ::= ( markupdecl | conditionalSect | DeclSep)*
```

Мы видим, что после XML-объявления могут следовать комментарии, инструкции обработки или же пустые пространства, но затем идёт Объявления типа документа, где «Name» — имя корневого тега, «ExternalID» — внешний идентификатор, а «intSubset» — объявление разметки или же ссылка на сущность. Как гласит спецификация, если внешний идентификатор объявляется вместе со внутренним объявлением, последнее идёт перед первым.

Например:

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

Здесь «SYSTEM "hello.dtd"» — внешний идентификатор: адрес «hello.dtd» позволяет задействовать данные в документе «hello.dtd» как объявления разметки.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

Здесь же разметка была объявлена местно.

1.5.3 Инструкция обработки

Инструкции обработки (англ. *processing instruction, PI*), позволяют размещать в документе инструкции для приложений. В следующем примере показана инструкция обработки, передающая xml-stylesheet-приложению (например, браузеру) инструкции в файле my-style.css посредством атрибута href:<?xml-stylesheet type="text/css" href="mystyle.css" ?>

```
<?xml-stylesheet href="my-style.css"?>
```

1.5.4 Комментарий

Комментарии (*англ. comment*) не относятся к символьным данным документа. Комментарий начинается последовательностью «<!--» и заканчивается последовательностью «-->», внутри не может встречаться комбинация символов «--». Символ & не используется внутри комментария в качестве разметки.

Пример:

```
<!-- это комментарий -->
```

1.6 Корневой элемент

1.6.1 Элемент и его разметка

Элемент *(англ. element)* является понятием логической структуры документа. Каждый документ содержит один или несколько элементов. Границы элементов представлены *начальным* и *конечным тегами*. Имя элемента в начальном и конечном тегах элемента должно совпадать. Элемент может быть также представлен тегом *пустого*, то есть не включающего в себя другие элементы и символьные данные, *элемента*.

Тег (*англ. tag*) — конструкция разметки, которая содержит имя элемента.

Начальный тег: <element1>

Конечный тег: </element1>

Тег пустого элемента: <empty_element1 />

В элементе атрибуты могут использоваться только в начальном теге и теге пустого элемента.

Пример кулинарного рецепта, размеченного с помощью XML:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE recipe>
<recipe name="хлеб" preptime="5min" cooktime="180min">
  <title>
    Простой хлеб
  </title>
  <composition>
    <ingredient amount="3" unit="стакан">Мука</ingredient>
    <ingredient amount="0.25"
unit="грамм">Дрожжи</ingredient>
    <ingredient amount="1.5" unit="стакан">Тёплая
вода</ingredient>
    <ingredient amount="1" unit="чайная
ложка">Соль</ingredient>
  </composition>
  <instructions>
    <step>
      Смешать все ингредиенты и тщательно замесить.
    </step>
    <step>
      Закрыть тканью и оставить на один час в тёплом помеще-
нии.
    </step>
```

```
    <!--
      <step>
        Почитать вчерашнюю газету.
      </step>
       - это сомнительный шаг...
     -->
    <step>
      Замесить ещё раз, положить на противень и поставить в духовку.
    </step>
  </instructions>
</recipe>
```

1.6.2 Секция CDATA

Секция CDATA не является логической единицей текста. Секция может встречаться в любом месте документа, где синтаксис позволяет размещать символьные данные. Секция начинается <![CDATA[и завершается]]>. Между этой разметкой находятся символьные данные, символьные данные при этом включают символы < > & в их непосредственной форме.

1.7 Корректный документ

Корректный (*англ. well-formed*) документ соответствует всем общим правилам синтаксиса XML, применимым к любому XML-документу: правильная структура документа, совпадение имен в начальном и конечном теге элемента и т. п. Документ, который неправильно построен, не может считаться документом XML.

1.8Пространства имён

Поскольку в разных XML-документах могут встретится одни и те же имена тегов и их атрибутов, имеющие совершенно разный смысл, надо иметь возможность их как-то различать. Для этого имена тегов и атрибутов снабжают кратким префиксом, который отделяется от имени двоеточием. Префикс имени связывается с идентификатором, определяющим пространство имен (namespace). Все имена тегов и атрибутов, префиксы которых связаны с одним и тем же идентификатором, образуют одно пространство имен, в котором имена должны быть уникальны. Префикс и идентификатор пространства имен определяются атрибутом xmlns следующим образом:

<ns:root_element_name xmlns:ns = "http://URI_namespace">

В дальнейшем имена те- гов и атрибутов, которые мы

хотим отнести к пространству имен "http://URI_namespace", снабжаются префиксом ns, например:

<ns:city ns:type="город">Новосибирск</ns:city>.

Атрибут xmlns может появиться в любом элементе XML, а не только в корневом. Определенный им префикс можно применять в том элементе, в котором записан атрибут xmlns, и во всех вложенных в него элементах. Более того, в одном элементе можно определить несколько пространств имен. Во вложенных элементах пространство имен можно переопределить, связав префикс с другим идентификатором. Появление имени тега без префикса в документе, использующем пространство имен, означает, что имя принадлежит пространству имен по умолчанию. Префиксы, начинающиеся с символов xml с любым регистром букв, зарезервированы за самим языком XML.

Имя вместе с префиксом называется расширенным или уточненным именем. Часть имени, записанная после двоеточия, называется локальной частью имени.

Идентификатор пространства имен должен иметь форму URI. Адрес URI не имеет никакого значения и может не соответствовать никакому действительному адресу Интернета. В данном случае URI можно рассматривать как уникальную строку символов, идентифицирующую пространство имен.

По правилам SGML и XML, двоеточие может применяться в именах как обычный символ, поэтому любая программа, "не знающая" пространства имен, анализируя документ, рассматривает уточненное имя как обычное уникальное имя. Отсюда следует, в частности, что в объявлении типа документа (Document Type Declaration) нельзя опускать префиксы имен.

1.9 Пример документа

Рассмотрим пример XML-документа. В XML можно назначить некоторое значение тегам в документе. Что еще более важно, эту информацию также легко обработать для машины. Можно выделить почтовый код из этого документа просто найдя содержимое, обрамленное тегами <postal-code> и </postal-code>, называемое элементом <postal-code>.

```
<address>
 <name>
  <title>Mrs.</title>
  <first-name>
   Mary
```

```
  </first-name>
  <last-name>
   McGoon
  </last-name>
 </name>
 <street>
  1401 Main Street
 </street>
 <city>Anytown</city>
 <state>NC</state>
 <postal-code>
  34829
  </postal-code>
</address>
```

2. РЕГЛАМЕНТАЦИЯ РАБОТЫ С ДОКУМЕНТАМИ: ПРАВИЛА, ЯЗЫКИ, ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ

Этот раздел содержит изложение некоторых положений рекомендаций W3C, касающихся работы с документами. Соответствующие рекомендации могут относиться как к документам XML, так и к более широкому классу документов. Ссылки, как правило, даются на средства работы с документами, рекомендованные W3C.

2.1 Кодировка документов

Спецификация требует, чтобы обрабатывающие программы поддерживали по крайней мере две кодировки Юникод: UTF-8 и UTF-16.

2.2 XML-процессор и приложение

Спецификация XML определяет понятия *XML-процессор* и *приложение*. XML-процессор (парсер) — программа, анализирующая разметку и передающая информацию о структуре документа другой программе — приложению.

Спецификация XML налагает определённые требования на процессор, не касаясь требований к приложению.

2.3 Действительный документ. Проверяющие и непроверяющие процессоры

Документ является *действительным*, если с ним связано объявление типа документа и если этот документ отвечает представленным в объявлении типа ограничениям.

XML процессоры делятся на два класса: проверяющие и непроверяющие.

Проверяющие процессоры проверяют действительность документа и должны сообщать (по выбору пользователя) о нарушении ограничений, сформулированных в объявлении типа документа.

Непроверяющие процессоры не проверяют действительность документа, но обязанности по предварительной обработке документа, упомянутые выше, остаются за ними.

2.4 Описание типов: языки схем

Для описания типов документов используются языки схем (англ. *schema language*). Поскольку XML является подмножеством языка SGML, то он унаследовал разработанный для SGML язык

Document Type Definition (DTD). Позднее были разработаны и другие языки схем, наиболее известны из которых XML Schema, RELAX NG.

2.5 Преобразование документа XML

Для решения задачи преобразования документа XML в другую схему или другой формат предназначен язык XSLT.

2.6 Формат для визуализации документа

Для форматированного документа (документа, подготовленного к визуализации) предназначен формат XSL-FO.

2.7 Языки запросов

XPath — синтаксис для адресации содержимого документа, представленного в форме дерева. Выражения XPath используются в языке *XQuery*. Выражения XPath, вообще говоря, могут использоваться в любом контексте, где уместно использовать формальные ссылки на элементы дерева, в частности, в качестве параметров для методов интерфейсов доступа к документу.

XQuery — язык программирования, ориентированный на работу с документами.

2.8 Чтение XML: три варианта API

Для чтения XML есть три варианта API.

Событийный API — XML-процессор читает XML; при определённом событии (появлении открывающего или закрывающего тега, текстовой строки, атрибута) вызывается callback-функция. Для простоты программирования событийный процессор может собрать в памяти все атрибуты одного тега на манер DOM — но это уже подразумевает, что пользователь «невраждебен» (неспособен или бессмысленно давать XML с большим количеством атрибутов в одном теге, чтобы заполнить всю память).

- \+ Быстр, расходует мало памяти.
- – Крайне сложен в программировании: приходится держать в памяти информацию, в каком месте документа мы находимся.
- \+ Библиотека проста в программировании.
- – «Почти верные» XML с перепутанным порядком тегов считаются ошибочными.
- – Если в XML много объектов с перекрёстными ссылками друг на друга, надо организовать временное хранение строковых ссылок, чтобы потом, когда документ

будет считан, преобразовать идентификаторы в указатели.

- – При ошибке в XML в памяти остаётся полусозданная структура предметной отрасли; программист должен своими руками корректно уничтожить её.
- Примеры библиотек: SAX

Потоковый API — устроен на манер потоков ввода-вывода. Как и в событийном API, для простоты программирования процессор может собирать в памяти все атрибуты одного тега на манер DOM — но это подразумевает, что пользователь «невраждебен».

- + Быстр, расходует мало памяти.
- – Довольно сложен в программировании. Впрочем, информация, в каком месте документа мы находимся, неявно задаётся местом в потоке выполнения.
- – Библиотека сложна в программировании.
- – «Почти верные» XML с перепутанным порядком тегов считаются ошибочными.
- – Если в XML много объектов с перекрёстными ссылками друг на друга, надо организовать временное хранение строковых ссылок, чтобы потом, когда документ будет считан, преобразовать в указатели. Продвинутые библиотеки могут запомнить внутреннее состояние процессора, а потом вернуться к нему; тогда можно реализовать простановку ссылок на манер DOM (первым проходом построить объекты без ссылок, вторым — восстановить ссылки). Правда, в таком случае XML считывается дважды, что медленно.
- – При ошибке в XML в памяти остаётся полусозданная структура предметной отрасли; программист должен своими руками корректно уничтожить её.
- Примеры библиотек: StAX

Объектный API (Document Object Model, DOM, «объектная модель документа») — считывает XML и воссоздаёт его в памяти в виде объектной структуры.

- – Наиболее медленный вариант; расходует много памяти. С учётом накладных расходов на объекты на x86 предельная длина файла — несколько сотен мегабайт.
- + Прост в программировании.
- + Библиотека проста в программировании.
- + Зачастую удаётся распознать «почти верные» XML с перепутанным порядком тегов.

- \+ Если в XML много объектов с перекрёстными ссылками друг на друга, достаточно дважды пройтись по документу: первый раз создать объекты без ссылок и заполнить словарь «название-объект», второй раз — восстановить ссылки.
- \+ При ошибке в XML в памяти остаётся полусозданная структура XML, которая будет автоматически уничтожена самой библиотекой.
- \+ Естественный выбор, когда объектом предметной области является сам XML: например, в веб-браузере, XML-редакторе, в импортёре к программе-локализатору, который извлекает строки из XML произвольной структуры.
- Примеры библиотек: JDOM, TinyXML

Бывают и гибридные API: внешние и маловажные части читаются потоковым методом, а внутренние и важные — объектным.

2.9 Запись XML: два варианта API

API прямой записи записывает XML тег за тегом, атрибут за атрибутом.

- \+ Быстр, нет промежуточных объектов.
- – Примитивная библиотека может делать неоптимальный XML (например, <tag></tag> вместо <tag />).
- – Непригоден для отдельных специфических задач.
- – Если структуры предметной области работают ненадёжно, без специальных мер (записать в память или в другой файл, потом переименовать) можно остаться с «упавшей» программой и потерянным файлом.
- – При ошибке программиста может получиться синтаксически некорректный XML.

Объектный API, он же Document Object Model.

- – Создаёт объектную структуру для XML, что может отнять памяти больше, чем структура предметной отрасли.
- \+ Универсален (впрочем, в большинстве задач преимущества над хорошо проработанным API прямой записи нет — в отличие от чтения).
- \+ Даже если структуры предметной области работают ненадёжно, а программист не предусмотрел никакой «защиты», единственный сценарий, когда файл перезаписывается на неполный — нехватка места на диске.

3. ИНСТРУМЕНТЫ РАБОТЫ С ДОКУМЕНТАМИ: ПАРСЕРЫ, СРЕДСТВА СОЗДАНИЯ И ВИЗУАЛИЗАЦИИ, СИСТЕМЫ БАЗ ДАННЫХ

3.1 Реализации парсеров

XML имеет реализации парсеров для всех современных языков программирования.

3.2 Веб-браузеры как инструмент визуализации документа

3.2.1 Визуализация без использования стилей CSS

Без использования CSS или XSL XML-документ отображается как простой текст в большинстве веб-браузеров. Некоторые браузеры, такие как Internet Explorer, Mozilla Firefox и Opera (встроенный инструмент Opera Dragonfly) отображают структуру документа в виде дерева, позволяя сворачивать и разворачивать узлы с помощью нажатий клавиши мыши.

3.2.2 Применение стилей CSS

Процесс аналогичен применению CSS к HTML-документу для отображения. Для применения CSS при отображении в браузере, XML-документ должен содержать специальную ссылку на таблицу стилей. Например:

<?xml-stylesheet type="text/css" href="myStyleSheet.css"?>

Это отличается от подхода HTML, где используется элемент <link>.

3.2.3 Применение преобразований к XS-FO формату

Современные браузеры принадлежат к числу программных средств, способных выполнять преобразования XSLT. В браузере такое преобразование выполняется, как правило, для форматирования документа (преобразования документа в формат XSL-FO). Следующая инструкция в прологе документа XML предписывает браузеру выполнить XSLT-преобразование, описанное в файле transform.xsl:

<?xml-stylesheet type="text/xsl" href="transform.xsl"?>

3.3 Редакторы XML

С документом XML можно работать в обычном текстовом редакторе, однако обычные редакторы не поддерживают структуру документа. Существуют специальные редакторы XML, которые делают работу с документом более удобной и эффективной.

3.4 Системы управления базами данных, работающие с данными в формате XML

Система управления базами данных DB2 позволяет хранить данные в формате XML и предоставляет доступ к таким данным с помощью языка XQuery.

3.5 Поддержка на аппаратном уровне

XML поддерживается на низком аппаратном, микропрограммном и программном уровнях в современных аппаратных решениях.

4. ОБЛАСТЬ ПРИМЕНЕНИЯ, ОГРАНИЧЕНИЯ, ПЕРСПЕКТИВЫ РАЗВИТИЯ

4.1 Эффективность использования XML

XML — язык разметки, другими словами, средство описания документа. Именно в нише документов, текстов, где доля разнотипных символьных данных велика, а доля разметки мала — XML успешен. С другой стороны, обмен данными в открытых системах не сводится к обмену документами. Избыточность разметки XML (а в целях разработки языка прямо указано, что лаконичность не является приоритетом проекта) сказывается в ситуациях, когда данные не вписываются в традиционную модель документа. Лента новостей, например, оформляемая с использованием синтаксиса XML (форматы RSS, Atom), представляет собой не документ в традиционном понимании, а поток однотипных мини-документов — многословная и избыточная разметка в этом случае составляет существенную часть передаваемых данных.

W3C озабочен эффективностью применения XML, и соответствующие рабочие группы занимаются этой проблемой (к началу 2013 года нормативные документы не разработаны).

Другая ситуация, когда форматы XML могут оказаться не лучшим решением — работа с данными с простой структурой и небольшим по объёму содержанием полей данных. В этом случае доля разметки в общем объёме велика, а программная обработка XML может оказаться неоправданно затратной, по сравнению с работой с данными более простой структуры. В этой области разработчики рассматривают средства, изначально ориентированные на данные, такие как INI, YAML, JSON.

4.2 Скриптовый язык для работы с XML

W3C работает над созданием скриптового языка для работы с XML (к началу 2013 года нормативные документы не разработаны).

ЗАКЛЮЧЕНИЕ

Усвоив несколько простых правил, вы можете гибко разрабатывать собственные элементы XML и их атрибуты. Правила XML не сложны. Набирать XML-документ тоже несложно. Главное - понять, что вы хотите от документов в смысле возможностей сортировки и поиска, а затем разработать элементы и атрибуты для удовлетворения этих требований.

Когда хорошо понимаешь цель и знаешь, как разметить свой текст, можно создавать эффективные элементы и атрибуты. С этой точки зрения тщательная разметка — это все, что нужно для создания правильно построенного и пригодного к использованию документа XML.

ЛИТЕРАТУРА

1. Дэвид Хантер, Джефф Рафтер, Джо Фаусе XML. Базовый курс. — М.: «Диалектика», 2019. — 1344 с.
2. Роберт Тейбор. Реализация XML Web-служб на платформе Microsoft .NET = Microsoft .NET XML Web Services. — М.: Вильямс, 2002. — 464 с.
3. Дэвид Хантер, Джефф Рафтер, Джо Фаусетт, Эрик ван дер Влист, и др. XML. Работа с XML, 4-е издание = Beginning XML, 4th Edition. — М.: «Диалектика», 2009. — 1344 с.