Структура XML-документов

По материалам курса University of Washington
http://www.cs.washington.edu/education/courses/cse190m/07sp/index.shtml

XML

Язык XML основан на HTML, но является, с одной стороны, его расширением, а с другой стороны, он имеет более строгие правила.

Основные цели создания XML:

1. Обеспечить возможность универсального текстового представления структурированных данных.
2. Дать возможность создавать языки, подобные XHTML на основе единых правил и с помощью простых средств.
3. Обеспечить возможность обработки структурированных данных с помощью универсальных программных средств.

Таким образом, с помощью XML удается:

1. Переносить данные из одной системы в другую; единственное условие – обе системы должны уметь разбирать структуру данных XML.
2. Просто определять легко понимаемые человеком и машиной языки, такие как XHTML, MathML, SVG и т.д.
3. Создавать универсальные программы обработки текстов на XML для получения внутреннего представления представленных структур (DOM).

Структура документа XML

В целом структура XML-документа знакома нам по структуре XHTML-документа: заголовок и тело, которое содержит корневой элемент и вложенные в него и друг в друга другие элементы.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element1 attr1="value1" attr2="value2">
    <element2>Содержание   моего     элемента</element2>
  </element1>
  <element3 attr1="value1" />
</root>
```

Замечания:

1. В 2006 году разработан стандарт XML 1.1, однако, в качестве номера версии нужно указывать 1.0, если только вы не используете в документе специфических возможностей XML 1.1.
2. Все программы обязаны понимать кодировки `UTF-8` и `UTF-16`, однако, если в вашем документе используются, скажем, русские символы – используйте соответствующую кодировку, скажем, `windows-1251`.
3. В отличие от HTML все пробелы и переводы строк внутри текстовых элементов сохраняются, а имена элементов и атрибутов чувствительны к регистру букв.

Проверка правильности XML-документа

Имеются две формы проверки правильности:

- формальная правильность (структурная); говорят, что формально правильный документ “well-formed”;
- содержательная правильность (по составу и содержанию тегов элементов и атрибутов); говорят, что содержательно правильный документ “valid”.

- для проверки формальной правильности достаточно проанализировать внешнюю структуру документа: правильно ли сделана вложенность элементов; все ли значения атрибутов заключены в кавычки и т.п.
- для проверки содержательной правильности используются специальные языки, определяющие допустимость тегов, атрибутов, правильность их вложенности друг в друга и т.п. Наиболее известным из таких языков является язык описания документов DTD.

Пример XML-документа

```
<?xml version="1.0" encoding="windows-1251"?>
<mail>
  <to name="Teacher@google.com"/>
  <from name="Student@mail.ru"/>
  <subject>How to get a best mark</subject>
  <content>Oh, please, please, help me!</content>
</mail>
```

Для форматирования XML-документа можно использовать язык CSS, только привязка CSS-страниц к XML-документу производится чуть-чуть по-другому, чем в HTML:

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

Однако, чаще для форматирования применяется другой способ: XML-документ преобразуется в HTML и форматируется с помощью XSL-документа. XSL – это отдельный сложный язык.

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

Использование «сущностей» (entities)

Если в текстовых элементах документа нужно использовать специальные символы <, >, &, ', ", то, как и в HTML вместо них используются замещающие их последовательности символов.
Набор стандартных сущностей невелик (только для пяти вышеперечисленных символов, нет даже сущности для «неразрываемого пробела»), однако при определении своего языка (с помощью DTD) можно определить свои собственные сущности.

Символ	«Сущность»
<	&lt;
>	&gt;
&	&amp;
' (апостроф)	&apos;
" (двойная кавычка)	&quot;
произвольный символ	&#<16-ичный код символа в указанной в заголовке кодировке>;

Использование неанализируемого текста

Если в тексте часто используются знаки &, <, >, то использовать сущности становится очень неудобно. Можно явно указать, что некоторый фрагмент текста не нужно анализировать на наличие сущностей и вложенных элементов, а нужно брать «как есть». Для этого используется специальный элемент CDATA:

```
<?xml version="1.0" encoding="UTF-8"?>
<function>
  <name>myFunction</name>
  <argument type="int" name="x"/>
  <body><[CDATA[
     if (x > 0 && x <= 10) {
       alert('invalid interval!');
     }
  ]]></body>
</function>
```

Описание конкретного языка с помощью DTD

Конкретное содержание документа может быть определено, если определен синтаксис языка, то есть состав элементов и атрибутов, возможные значения атрибутов и т.п. Это можно сделать с помощью языка описания структуры XML-документа DTD.

Как обычно, начинаем со случая, когда описание структуры документа вложено непосредственно в сам документ («смешивание языков»).

Описание структуры документа начинается с описания структуры корневого узла, которое содержит описание элементов, атрибутов и т.д.

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE root [ описание элементов ]>
<root>
  содержание документа
</root>
```

Простой пример DTD

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
  <to>Студенты</to>
  <from>Преподаватель</from>
  <heading>Заметка</heading>
  <body>Не забудьте про экзамен!</body>
</note>
```

В этом примере описано, что корневой элемент должен содержать элементы `to`, `from`, `heading` и `body` (в указанном порядке), а, в свою очередь, эти элементы могут содержать внутри себя произвольный текст.

Вынесение DTD в отдельный документ

Описание типа документа может быть отделено от самого документа. Описание элементов тогда будет содержаться в отдельном файле, а из основного документа будет ссылка на него.

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE note SYSTEM "myDTD.dtd">
<note>
  <to>Студенты</to>
  <from>Преподаватель</from>
  <heading>Заметка</heading>
  <body>Не забудьте про экзамен!</body>
</note>
```

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to      (#PCDATA)>
<!ELEMENT from    (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body    (#PCDATA)>
```

Структура блоков XML-документа

Синтаксически XML-документ состоит из следующих строительных блоков:

1. Элементы (elements)
2. Атрибуты (attributes)
3. Сущности (entities)
4. Текст PCDATA (parsed character data)
5. Текст CDATA (character data)

Примеры из XHTML:

1. Элементы: `p в <p>Это просто параграф</p>`
2. Атрибуты: `src в <img src="mypicture.jpg"/>`
3. Сущности: `nbsp в <p>Слова отделены пробелами</p>`
4. Текст PCDATA: текст, в котором происходит перевод сущностей и анализируются вставленные элементы (как внутри параграфа).
5. Текст CDATA: текст, который берется без изменений (как в <pre>)

Описание элементов XML-документа

При описании элементов указывают, что может содержаться внутри элемента, а также, какие он может иметь атрибуты. Вот несколько способов описать содержимое:

```
<!ELEMENT br EMPTY>
```

Элемент должен быть пустым

```
<!ELEMENT subject (#PCDATA)>
```

Элемент должен содержать произвольный текст

```
<!ELEMENT message ANY>
```

Элемент может содержать произвольную комбинацию текста и любых других элементов.

```
<!ELEMENT note (to,from,subject,body)>
```

Элемент должен содержать указанные элементы в одном экземпляре в указанном порядке. Все дочерние элементы должны быть также описаны.

Описание элементов (продолжение)

```
<!ELEMENT note (message+)>
```

Элемент должен содержать один или более элементов `message`.

```
<!ELEMENT note (message*)>
```

Элемент должен содержать произвольное число элементов `message`, в том числе и ни одного.

```
<!ELEMENT note (message?)>
```

Элемент может содержать не более одного элемента `message`.

```
<!ELEMENT note (to,(message|body))>
```

Элемент должен элемент `to`, а после него – `message` или `body`.

```
<!ELEMENT note (to|from|subject|body|#PCDATA)*>
```

Элемент может содержать любые из указанных элементов или текст в произвольном порядке в любом количестве.

Описание атрибутов

Предназначено для описания состава и содержания атрибутов некоторого элемента и имеет следующий вид:

```
<!ATTLIST element-name attribute-name attribute-type
          default-value>
```

Например, если элемент `schedule` может содержать атрибут `date` как показано ниже:

```
<schedule date="13.06.2008">
```

то описание синтаксиса может выглядеть так:

```
<!ATTLIST schedule date CDATA #REQUIRED>
```

Типы и значения атрибутов

Вот несколько возможных типов атрибутов (список не полный):

`CDATA`	Произвольный текст
`(val1|val2|...)`	Одно из указанных значений
`ID`	Уникальный в документе идентификатор
`IDREF`	Ссылка на другой идентификатор в документе
`ENTITY`	Сущность

Вот несколько возможных способов задать значение атрибута по умолчанию:

`value`	Указано конкретное текстовое значение
`#REQUIRED`	Обязательный атрибут
`#IMPLIED`	Необязательный атрибут
`#FIXED value`	Значение атрибута должно быть только таким, как указано

Атрибуты и вложенные элементы

Атрибуты, как и вложенные элементы, можно использовать для описания структуры элементов. Иногда трудно выбрать, какой способ предпочесть.

```
<mail date="25/12/2008">
  <to>friends</to>
</mail>
```

или

```
<mail>
  <date>25/12/2008</date>
  <to>friends</to>
</mail>
```

или, может быть,

```
<mail>
  <date>
    <day>25</day><month>12</month><year>2008</year>
  </date>
  <to>friends</to>
</mail>
```

Основное правило: используйте атрибуты только если

- значение точно не будет иметь внутренней структуры;
- значение уточняет информацию об элементе (мета-данные).

Описание сущностей

Сущности используются для введения часто используемых текстовых фрагментов и символов. Синтаксис:

```
<!ENTITY entity-name "entity-value">
```

например,

```
<!ENTITY copyright "Google 2008 and onwards">
<!ENTITY open "&lt;&lt;">
<!ENTITY close "&gt;&gt;">
```