Определения типа документа

DTD определяет, в каких отношениях между собой должны состоять элементы внутри документа XML. DTD содержит также правила грамматик и для документа и для всех его элементов. Документ, придерживающийся спецификаций, содержащихся в своем DTD, считается действительным (valid). (Не путайте с правильно построенным (well-formed) документом, в котором соблюдаются синтаксические правила XML.)

Объявления элементов

Все элементы, помещаемые в документ XML, должны быть объявлены в DTD. Это можно сделать с помощью объявления <!ELEMENT>, использующего следующий формат:

```
<!ELEMENT имя_элемента правило>
```

Эта строка определяет элемент XML и связанное с ним правило, которое логически объединяется с элементом в документе XML. В имени элемента не допускается использование символов <>. Имя должно начинаться с буквы или символа подчеркивания, за которыми может следовать любое количество букв, цифр, дефисов, точек и символов подчеркивания. Имена элементов не могут начинаться со строки xml в любом сочетании верхнего и нижнего регистров. Двоеточие в имени элемента допускается только для указания, пространства имен, в других случаях его использование запрещено.

ANY и PCDATA

Простейшее объявление элемента указывает, что между открывающим и закрывающим тегами элемента может находиться любое содержание, например,

```
<!ELEMENT library ANY>
```

Использование ключевого слова ANY позволяет включать как теги, так и символьные данные между элементами. Однако может потребоваться, чтобы допускались только символы. Данные такого типа называют анализируемыми символьными данных (parsed character data) или, кратко, PCDATA. Потребовать, чтобы элемент содержал только PCDATA, можно, например, с помощью следующего объявления:

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

Это объявление означает, что между тегами элемента могут находиться любые символьные данные, которые не являются элементом. Поэтому допустима следующая запись в документе XML:

```
<title></title>
<tit1e>Проектирование баз данных на основе XML</tit1e>
<title>Программирование для SQL Server 2000 с использованием XML</title>
```

Однако предыдущее объявление PCDATA не позволяет написать:

```
<tit1e>Проектирование баз данных на основе <emphasis>XML</emphasis></tit1e>
```

Тем не менее, иногда возникает необходимость указать, что между двумя заданными тегами должен присутствовать другой элемент. Сделать это можно, поместив название элемента в скобки. Следующие два правила устанавливают, что элемент <books> должен, содержать элемент <title>, а элемент <title> должен содержать анализируемые символьные данные (или пустое содержимое), но не другой элемент:

```
<!ELEMENT books (title)>
<!ELEMENT title (#PCDATA)>
```

Множественные элементы

Если необходимо, чтобы несколько элементов содержались между открывающим и закрывающим тегами некоторого элемента в указанном порядке, можно использовать запятую (,) для разделения двух элементов:

```
<!ELEMENT books (title, authors)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authors (#PCDATA)>
```

В этом объявлении DTD требует, чтобы между открывающим тегом <books> и закрывающим тегом </books> присутствовал состоящий из анализируемых символьных данных элемент <title>, а сразу за ним следовал элемент <authors>, также состоящий из анализируемых символьных данных. Элемент <authors> не может предшествовать элементу <title>. Вот действительный документ XML для определенного выше фрагмента DTD:

```
<books>
<title>XML для разработчиков-профессионалов .NET</title>
<authors>Динар Дальви, Джо Грэй</authors>
</books>
```

В последнем примере показано, как определить в объявлении оба элемента. Столь же просто указать, что в документе должен появиться тот или иной элемент (но не оба сразу), используя вертикальную черту (|):

```
<!ELEMENT books (title | authors)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authors (#PCDATA)>
```

В этом DTD указано, что внутри элемента <books> может появиться либо элемент <title>, либо элемент <author>. Обратите внимание, что тот или другой элемент должен обязательно присутствовать. Если опустить оба элемента или оба включить, то документ XML не будет считаться действительным.

Группировка и повторяемость

Допускается вложенность скобок в объявлении для большей детализации определяемого синтаксиса. Например, приведенное ниже DTD указывает, что в элементе <books> документа XML должен содержаться либо элемент <description>, либо элемент <title> с непосредственно следующим за ним элементом <author>. Все три элемента должны состоять из анализируемых символьных данных.

```
<!ELEMENT books ((title, authors) | description)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authors (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

Теперь самое интересное: в объявлении элемента разрешается указывать, что некоторый элемент (или группа элементов в скобках) должны появиться один раз или ни разу, один или более раз или любое количество раз, включая ноль. Используемые для этого указания символы необходимо располагать сразу после целевого элемента (или группы элементов), к которому они относятся. Эти символы приведены в следующей таблице.

Атрибут	Описание
?	Должен появиться один раз или ни разу (0 или l раз)
+	Должен появиться хотя бы один раз (l или более раз)
*	Может появиться любое число раз или ни разу (0 или более раз)

Например, если необходимо более точно описать элемент <author>, можно назначить такое определение DTD:

```
<!ELEMENT author (authorname+)>
<!ELEMENT authorname (#PCDATA)>
```

Оно означает, что элемент <author> должен иметь внутри хотя бы один элемент <authorname>. Допускается также иметь несколько таких элементов. Более сложные взаимосвязи можно определить, используя скобки:

```
<!ELEMENT reviews (rating, synopsis?, comments+)*>
<!ELEMENT rating ((tutorial | reference)*, overall)>
<!ELEMENT synopsis (#PCDATA)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT tutorial (#PCDATA)>
<!ELEMENT reference (#PCDATA)>
<!ELEMENT overall (#PCDATA)>
```

Пустые элементы

Необходимо определить все пустые элементы, которые могут использоваться в действительном документе XML. Это можно сделать с помощью ключевого слова EMPTY:

```
<!ELEMENT имя-элемента EMPTY>
```

Например, в следующем объявлении определяется элемент документа XML, который можно использовать в виде `<statuscode/>` или `<statuscode></statuscode>`:

```
<!ELEMENT statuscode EMPTY>
```

Объекты

В DTD можно объявить *объект* (entity), который позволяет использовать *ссылку на объект* для замены одной последовательности символов на другую в документе XML, что аналогично использованию макроопределений.

Общие объекты

Общие объекты (general entities) – это объекты, которые могут служить заменой других символов в документе XML. Объявление общего объекта имеет следующий формат:

```
<!ENTITY имя "символы-замены">
```

Ранее мы уже видели пять общих ссылок на объекты для каждого из символов <, >, &, ‘ и “. Любую из них можно включать в документ XML, чтобы предотвратить интерпретацию этих символов процессором XML как разметки. (Кстати, объявлять их в своем DTD не нужно: они всегда доступны.)

В качестве примера можно привести ссылку на объект для символа `copyright`. Такой объект можно объявить в DTD с помощью строки

```
<!ENTITY copyright "&#xA9">
```

Объект `&copyright;` связан со значением Unicode 169 (или шестнадцатеричным 0xA9), являющимся символом @. После этого в документе XML можно писать:

```
<copyright>&copyright; 2006 моя корпорация</copyright>
```

Существует два ограничения для объявления объектов:

- Нельзя делать в объявлениях циклические ссылки. Например, следующие объявления недопустимы:

```
<!ENTITY entitya "&entityb; является по-настоящему правильным!">
<!ENTITY entityb "&entitya; также является по-настоящему правильным!">
```

- В DTD не допускается замена текста, не являющегося документом с помощью общей ссылки на объект. Разрешение общей ссылки на объект осуществляется только в документе XML, а не в документе DTD. (Если необходимо, чтобы разрешение ссылки на объект осуществлялось в DTD, нужно использовать ссылку на параметрический объект.)

Параметрические объекты

Ссылки на параметрические объекты могут присутствовать только в DTD и заменяются определениями своих объектов в DTD. Все ссылки на параметрические объекты начинаются со знака процента, означающего, что их нельзя использовать в документах XML – только в DTD, где они определены. Вот как объявляется параметрическая сущность:

```
<!ENTITY %имя "символы-замены">
```

Несколько примеров использования ссылок на параметрические объекты:

```
<!ENTITY %pcdata "(#PCDATA)">
<!ELEMET authortitle %pcdata;>
```

Обратите внимание, что для ссылок на параметрические объекты действуют так же два ограничения, что и для общих ссылок на объекты. Кроме того, ссылки на параметрические объекты нельзя использовать прежде их объявлений.

Внешние объекты

XML позволяет объявить внешний объект, например, так:

```
<!ENTITY quotes SYSTEM "http://www.oreilly.com/stocks/quotes.xml">
```

в результате становится возможным копирование содержимого XML, расположенного по указанному URI, в текущий документ XML с помощью ссылки на внешний объект. Например

```
<document>
<heading>Текущие рыночные котировки</heading>
&quotes;
</document>
```

В итоге, содержание XML, расположенное на URI http://www.oreilly.com/stocks/quotes.xml, копируется в документ при обработке его процессором XML. Нетрудно догадаться, что это удобно использовать при работе с динамическими данными.

Неанализируемые объекты

С помощью таких же опознавательных знаков можно использовать неанализируемые объекты для объявления в документе XML содержимого, не являющегося XML. Например, если нужно объявить внешнее изображение для использования в документе XML, можно указать в DTD следующее:

```
<!ENTITY image1 SYSTEM http://www.oreilly.com/ora.gif NDATA GIF89a>
```

Обратите внимание, что мы используется также ключевое слово `NDATA` (notation data - данные обозначений), точно указывающее на тип неанализируемых данных, с которым имеет дело процессор XML. Обычно ссылка на неанализируемый объект используется в качестве значения атрибута элемента, определяемого в DTD с типом `ENTITY` или `ENTITIES`. Вот пример использования только что объявленного неанализируемого объекта:

```
<image src="&image1;"/>
```

Нотации

Наконец, вместе с неанализируемыми объектами используются нотации. Объявление нотации просто сопоставляет значение ключевого слова NDATA, использованного выше, с более специфической информацией. Процессор XML может использовать эти данные, как считает нужным, или игнорировать их.

```
<!NOTATION GIF89a SYSTEM
"-/ /CompuServe/ /NOTATION Graphics Interchange Format 89a/ /EN">
```

Объявления атрибутов в DTD

Атрибуты для различных элементов XML должны быть указаны в DTD. Назначить атрибуты можно с помощью объявления <!ATTLIST> вида

<!ATTLIST *целевой_элемент имя_атрибута тип_атрибута значение_по_умолчанию*>

Объявление <!ATTLIST> состоит из имени целевого элемента, имени атрибута, типа его данных и значения, которое должно ему присваиваться по умолчанию. Вот несколько примеров допустимых объявлений <!ATTLIST>:

```
<!ATTLIST box length CDATA "0">
<!ATTLIST box width CDATA "0">
<!ATTLIST frame visible (true false) "true">
<!ATTLIST person marital (single | married | divorced | widowed) #IMPLIED>
```

В этих примерах первое слово после ATTLIST объявляет имя целевого элемента (т. е. box, frame, person). Затем следует имя атрибута (length, width, visible, marital). Дальше, как правило, указывается тип атрибута и его значение по умолчанию.

Значение по умолчанию

В случае, когда использование значения по умолчанию неуместно, можно вместо него указывать ключевые слова (модификаторы), перечисленные в таблице.

Атрибут	**Описание**
#REQUIRED	Значение атрибута должно обязательно быть указано для элемента
#IMPLIED	Значение атрибута может остаться незаданным
#FIXED	Значение атрибута фиксировано и не может быть изменено пользователем

Если используется ключевое слово #IMPLIED, и значение атрибута не определено и не установлено в документе XML, анализатор XML должен уведомить приложение, что значение не указано. Приложение может предпринять при этом действия, которые сочтет необходимыми. При использовании ключевого слова #FIXED непосредственно за ним должно быть определено значение по умолчанию, как показано в примере:

```
<!ATTLIST date year CDATA #FIXED "2006">
```

Типы данных

В таблице перечислены типы данных, которые можно использовать в DTD.

Атрибут	**Описание**
CDATA	Символьные данные
перечисление	Ряд значений, из которых можно выбрать только одно
ENTITY	Объект, определенный в DTD
ENTITIES	Несколько объявленных в DTD объектов, разделяемых знаками пробелов, табуляции, перевода строки
ID	Уникальный идентификатор элемента

IDREF	Значение атрибута типа уникальный ID
IDREFS	Несколько элементов IDREF
NMTOKEN	Именованная лексема XML
NMTOKENS	Несколько именованных лексем XML, разделяемых знаками пробелов, табуляции, перевода строки
NOTATION	Нотация, объявленная в DTD

Ключевое слово CDATA просто объявляет, что значением могут быть любые символьные данные. Ниже приведено несколько примеров объявлений атрибутов с использованием CDATA:

```
<!ATTLIST person name CDATA #REQUIRED>
<!ATTLIST person email CDATA #REQUIRED>
<!ATTLIST person company CDATA #FIXED "O'Reilly">
```

В случае перечисления ключевое слово не используется, а просто указывается само перечисление:

```
<!ATTLIST person marital (single | married | divorced | widowed) #IMPLIED>
<!ATTLIST person sex (male | female) #REQUIRED>
```

Типы данных ID, IDREF и IDREFS позволяют определить атрибуты как ссылки на один или несколько ID. ID – это просто атрибут, значение которого отличает данный элемент от всех остальных в текущем документе XML. ID полезны для ссылок на различные части документа, содержащие элемент с уникальным ID. Атрибуты IDREF служат для ссылки на другие ID. Рассмотрим, например, такой документ XML:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE sector SYSTEM sector.dtd>
<sector>
<employee empid="e1013">Джек Рассел</employee>
<employee empid="e1014">Сэмюэль Тессен</employee>
<employee empid="e1015" boss="e1013">Тьерри Уайт</employee>
<employee empid="e1016" boss="e1014">Стив Макалистер</employee>
</sector>
```

Его DTD выглядит следующим образом:

```
<!ELEMENT sector (employee*)>
<!ELEMENT employee (#PCDATA)>
<!ATTLIST employee empid ID #REQUIRED>
<!ATTLIST employee boss IDREF #IMPLIED>
```

Здесь каждый служащий (employee) имеет свой собственный идентификационный номер (e1013, e1014 и т. д.), который определен в DTD с помощью атрибута empid с ключевым словом ID. Этот атрибут образует идентификатор ID для каждого элемента <employee>. Никакие два элемента <employee> не могут иметь одинаковые ID.

В атрибутах, применяемых только для ссылки на другие элементы, используется тип данных IDREF. В этом случае атрибут boss имеет тип IDREF, поскольку он может принимать в качестве своего значения только значения других атрибутов ID. Роль ID станет более понятна при рассмотрении XLink и XPointer.

Атрибуты NMTOKEN и NMTOKENS объявляют имена именованных лексем XML. …
Эти типы данных полезны при перечислении названий языков или других наборов ключевых слов, следующих этим ограничениям в DTD.

ENTITY позволяет использовать объект, объявленный в DTD. Это может быть неанализируемый объект. Например, для ссылки на изображение можно сделать объявления:

```
<!ELEMENT image EMPTY>
<!ATTLIST image src ENTITY #REQUIREQ>
<!ENTITY chapterimage SYSTEM "chapimage.jpg" NDATA "jpg">
```

и использовать их следующим образом:

```
<image src="chapterimage">
```

Ключевое слово NOTATION означает, что ожидается нотация, созданная в DTD объявлением <!NOTATION>. В следующем примере атрибут player элемента <media> может принимать значение mpeg или jpeg:

```
<!NOTATION mpeg SYSTEM "mpegplay.exe">
<!NOTATION jpeg SYSTEM "netscape.exe">
<!ATTLIST media player NOTATION (mpeg | jpeg) #REQUIRED>
```

Обратите внимание, что необходимо перечислить все обозначения, допустимые в атрибуте. Например, для определения возможных значений атрибута player элемента <media> используйте такие записи:

```
<!NOTATION mpeg SYSTEM "mpegplay.exe")
<!NOTATION jpeg SYSTEN "netscape.exe")
<!NOTATION movie SYSTEM "mplayer.exe">
<!NOTATION avi SYSTEM "mplayer.exe">
<!ATTLIST media player NOTATIONS (mpeg | jpeg | movie) #REQUIRED>
```

Заметьте также, что, согласно правилам этого DTD, элементу <media> не разрешается воспроизводить файлы AVI.

Обратите внимание на то, что все объявления можно поместить в одно объявление ATTLIST, если следовать правилам для каждого типа данных:

```
<!ATTLIST person
name CDATA #REQUIRED
number IDREF #REQUIRED
company CDATA #FIXED "O'Reilly">
```

Включаемые и игнорируемые помеченные разделы

Внутри DTD можно связать в одну группу объявления, которые следует игнорировать, с помощью директивы IGNORE:

```
<![ IGNORE [
Игнорируемое_содержание_DTD
]]>
```

И, напротив, если нужно обеспечить включение объявлений в DTD, используется директива INCLUDE, имеющая аналогичный синтаксис:

```
<! [ INCLUDE [
Включаемое_содержание_DTD
]]>
```

Для чего может понадобиться использование этих объявлений, становится понятно, как только директивы INCLUDE или IGNORE заменяются ссылкой на параметрический объект, который можно легко изменить на месте. Рассмотрим, например, следующее DTD:

```
<!ENTITY %instruct "INCLUDE">
<! [ %ifsrict; [
<!ELEMENT reviews (rating, synopsis?, comments+)*>
<!FLEMENT rating ( (tutorial | reference)*, overall)>
```

```
<!ELEMENT synopsis (#PCDATA)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT tutorial (#PCDATA)>
<!ELEMENT reference (#PCDATA)>
<!ELEMENT overall (#PCDATA)>
]]>
```

Вложенные в это DTD объявления можно включить или удалить, присвоив параметрическому объекту `%ifstrict;` значение последовательности символов `"IGNORE"` или `"INCLUDE"`, вместо того чтобы заключать все объявления в комментарии, когда в них нет нужды.

Внутренние подмножества

Часть объявлений DTD можно поместить в объявление `DOCTYPE` документа XML, как показано ниже:

```
<!DOCTYPE b6i1erplate SYSTEM "generic_inc.dtd" [
<!ENTITY corpname "Acme, Inc.">
]>
```

Пространство между скобками называется внутренним подмножеством DTD. При чтении DTD синтаксическим ализатором сначала читается внутреннее подмножество, а затем внешнее подмножество, являющееся файлом, на который ссылается объявление `DOCTYPE`.
Существуют ограничения на сложность внутреннего подмножества, а также предположения обработки, влияющие на, то, как следует его структурировать:

- Во-первых, во внутреннем подмножестве не допускаются условные разделы, такие как `INCLUDE` или `IGNORE`.
- Во-вторых, все ссылки на параметрические сущности должны расширяться в ноль или более объявлений. Например, допустимо указывать ссылку на параметрический объект

```
%paradecl;
```

если `%paradecl;` раскрывается в

```
<!ELEMENT para CДATA>
```

Однако если просто записать во внутреннем подмножестве

```
<!ELEMENT para (%paracont;)>
```

это будет считаться недопустимым, поскольку не происходит расширения в полное объявление.

Предполагается, что анализатор, не осуществляющий проверки действительности внешнего подмножества (nonvalidating parser), все же обрабатывает значения атрибутов по умолчанию и объявления объектов во внутреннем подмножестве. Однако параметрический объект может изменить значение этих объявлений таким образом, что они станут неподдающимися анализу. Поэтому анализатор должен прекратить обработку внутреннего подмножества, дойдя до первой ссылки на внешний параметрический объект, который он не обрабатывает. Если это внутренняя ссылка, анализатор может раскрыть ее, а решение получить объект позволит ему продолжить обработку. Если он не обработает замену объекта, то обработка списка атрибутов и объявлений объектов во внутреннем подмножестве не должна проводиться.

Для чего это делается? Поскольку некоторые объявления объектов часто относятся только к одному документу, например, объявления объектов глав или других файлов содержимого, их разумно помещать во внутреннее подмножество. Аналогично, если для некоторого документа требуется переопределить или изменить используемые значения DTD, можно поместить во внутреннее подмножество новое определение. Наконец, если процессор XML не проверяет внешнее подмножество (о чем мы говорили выше), то внутреннее подмножество будет самым лучшим местом для помещения некоторых связанных с DTD данных, таких как идентификация атрибутов `ID` и `IDREF`, значений атрибутов по умолчанию и объявлений объектов.