# Формирование действительных XML-документов. DTD

Каждый XML-документ должен быть *правильно построенным* (well-formed), т.е. отвечать минимальным требованиям по составлению XML-документов. Иначе он не может считаться XML-документом.

*Действительным* (valid) называется корректно сформированный правильно построенный документ, отвечающий двум дополнительным требованиям:

* структура документа должна быть задана в виде схемы (DTD, XML Schema и/или др.), которая объявляется в прологе XML;
* тело документа должно соответствовать заданной структуре.

Зачем нужна схема документа? Схема документа вносит строгость и точность в правила написания документов XML.

Создавая XML-документ, разработчик сам решает: как назвать теги, в каком порядке они будут следовать, какие данные в них будут записаны, будут ли у элемента атрибуты и т.д.

Без формального описания структуры документа этим самым документом может воспользоваться только его разработчик. Если же XML-документ предназначен для передачи во внешний мир, без схемы документа не обойтись. Схема позволяет всем пользователям XML понимать и использовать элементы и атрибуты одинаково.

## DTD-схема документа

**DTD** (Document Type Definition — **определение типа документа**) – «классическая» схема документа, предложенная еще для SGML.

Определение типа документа описывает его информационную структуру. В DTD перечисляются все возможные имена элементов, определяются их атрибуты и описывается вложенность элементов.

DTD по определению содержит всю информацию, которая может появиться в XML-документе. Все, что входит в проект, должно быть включено в DTD.

Основная задача DTD – это проверка готового XML-документа на наличие ошибок. Пропущенные ошибки могут вызвать повреждение программы обрабатывающей данные документы, либо ввод в систему неверных данных.

## Ассоциирование DTD с документом XML

Для связывания декларации DTD с экземпляром документа в версии XML 1.0 предлагается специальная декларация DOCTYPE. Она должна следовать после декларации XML и предшествовать любым элементам документа. Тем не менее, между декларациями XML и DOCTYPE могут находиться комментарии и команды обработки.

Декларация DOCTYPE содержит ключевое слово DOCTYPE, за которым следует имя корневого элемента документа, а затем конструкция с декларациями содержания. Первые три строчки документа XML:

<xml version ="1.0" ?>

<!DOCTYPE root ... >

<root > ...

DTD может быть описан непосредственно в DOCTYPE (*внутренняя* декларация) или в отдельном файле формата \*.DTD (*внешняя* декларация). *Декларация* иначе называется *объявлением*. Возможен и *смешанный* вариант.

Декларация XML может содержать атрибут standalone, принимающий только значения "yes" и "nо". Если значение атрибута равно yes, то внешние декларации не влияют на информацию, передаваемую из документа в использующее его приложение. Значение no показывает, что существуют внешние декларации, необходимые для правильного описания документа.

На практике необязательный атрибут standalone используется редко. Наличие этого атрибута со значением, yes не гарантирует отсутствия внешних зависимостей любого типа. Просто внешние зависимости в этом случае не приведут к ошибке. Таким образом, в основном этот атрибут представляет собой знак для анализаторов и других приложений, показывающий, нужно ли им использовать какое-либо внешнее содержание.

### Внутренняя декларация DTD

В простейшем случае DTD можно внедрить непосредственно в XML-документ, расположив его внутри декларации DOCTYPE.

Блок внутренней декларации разметки тега DOCTYPE состоит из левой квадратной скобки, списка деклараций и правой квадратной скобки:

<! DOCTYPE root [

...здесь находятся

определения DTD ...

]>

Достоинство:

* всегда содержатся в документе, поэтому не возникает проблем с доступом к DTD – их нельзя «потерять».

Недостатки:

* нужно повторять в каждом документе;
* увеличение размера xml-файла;
* возможны расхождения в схемах разных документов и, как следствие, возникновение ошибок.

### Внешняя декларация DTD

В данном случае декларация DOCTYPE состоит из обычного ключевого слова и имени корневого элемента, за которым следует еще одно ключевое слово SYSTEM либо PUBLIC, обозначающее источник внешнего определения DTD, а за ним – локализация этого определения.

Если используется ключевое слово SYSTEM, DTD обязано находиться непосредственно по указанному адресу:

<! DOCTYPE root SYSTEM "myschema.dtd">

Ключевое слово PUBLIC предназначено для общеизвестных схем, для которых нет постоянного места хранения DTD. Приложение, анализирующее такой документ, должно использовать некоторый алгоритм для поиска DTD:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

Стандарт XML 1.0 допускает у декларации PUBLIC наличие как публичного URI, так и системного идентификатора. Если работающее с документом приложение или анализатор не может найти DTD по URI, оно должно использовать системный идентификатор.

Внешние DTD позволяют сократить объем хранимой информации, избежать множества ошибок и унифицировать все документы, использующие данный DTD-файл.

### Смешанный вариант

На практике внутренние DTD чаще всего применяются одновременно с внешними. Dо внутренних DTD могут быть заданы новые декларации или переписаны те, что содержатся во внешних.

Если какой-либо объект объявлен одновременно во внешнем и внутреннем определении DTD, приоритетом пользуется внутреннее объявление. Это позволяет приспособить DTD для конкретных документов.

Однако, если внешние DTD переписываются очень часто, они начинают терять свое значение, а это признак плохого первоначального проекта.

## Основные определения

Допустимое в документе XML содержание определяется с помощью четырех типов определений. В следующей таблице показаны ключевые слова объявлений и их значения.

|  |  |
| --- | --- |
| **Конструкция DTD** | **Значение** |
| ELEMENT | Объявление элемента XML |
| ATTLIST | Объявление атрибутов для ранее объявленных элементов, а также разрешенных значений этих атрибутов |
| ENTITY | Объявление многократно используемого содержания |
| NOTATION | Объявление форматирования для внешних данных, которые не должны анализироваться (например, двоичные данные) |

Первые два связаны с информацией, которую мы рассчитываем найти в документе XML, – элементами и атрибутами.

Сущности (ENTITY) используются для поддержки, чтобы облегчить задачи разработчика. Как правило, они состоят из содержания, которое часто используется в DTD или документе, или которое необходимо изменять одновременно во всех документах. Например, часто используемый фрагмент текста, дата последнего обновления. Их употребление подобно константам в языках программирования.

Инструкции по обработке (NOTATION) описывают содержание, разработанное не на языке XML. Используются они для того, чтобы объявить конкретный класс данных и связать его с внешней программой. Эта внешняя программа становится обработчиком объявленного класса данных. Например, связав с документом изображение в формате JPEG, разработчик желает, чтобы программа приняла и отобразила двоичные данные в этом формате.

## Объявление элементов

Каждый элемент документа XML должен быть описан. Объявление элемента начинается с символов <!ELEMENT, после которых через пробел идет имя элемента и его содержимое. Заканчивается объявление символом >. По своему содержанию элементы делятся на четыре группы.

1. Пустой элемент – может иметь атрибуты, но не содержит текст или дочерние элементы.

Объявление: после имени элемента указывается ключевое слово EMPTY. Пример:

<!ELEMENT element\_name EMPTY>

1. Элемент содержит только дочерние элементы, но не текст.

Объявление: после имени элемента в скобках через запятую перечисляются все вложенные элементы. Причем вложенные элементы должны следовать в XML-документе строго в том порядке, в каком они перечислены в объявлении. Пример:

<ELEMENT element\_name (elem\_1,elem\_2)>

1. Элемент содержит текст и, возможно, дочерние (вложенные) элементы.

Объявление: после имени элемента в скобках указывается ключевое слово #PCDATA, после которого через запятую, как и в предыдущем случае, перечисляются все вложенные элементы (если они имеются). Пример:

<ELEMENT element\_name (#PCDATA, elem\_1,elem\_2)>

<ELEMENT element\_name (#PCDATA)>

1. Элемент, открытый для любого содержания.

Объявление: после имени элемента указывается ключевое слово ANY. Пример:

<ELEMENT element\_name ANY>

Иногда из нескольких дочерних элементов разрешается использовать только один. В таком случае их имена перечисляются через вертикальную черту( | ). Например:

<!ELEMENT element\_name (elem\_1,(elem\_2|elem\_3))>

Элемент element\_name должен содержать элемент elem\_1, а затем либо elem\_2, либо elem\_3. Элементы появляются именно в таком порядке.

Если вложенный элемент можно записать в объявляемом элементе несколько раз, то необходимо указать его кратность сразу после имени (без пробелов):

? – элемент или список может встретиться 0 или 1 раз;

\* – элемент или список может встретиться 0 или более раз;

+ – элемент или список может встретиться 1 или более раз.

Например:

<!ELEMENT element\_name (elem\_1,elem\_2\*,elem\_3?, elem\_4+)>

Первым всегда объявляется корневой элемент. После него в соответствующем порядке идут дочерние элементы и их атрибуты.

## Объявление атрибутов

Все атрибуты элемента объявляются сразу после объявления самого элемента, одним списком.

Список начинается с <!ATTLIST, затем через пробел следует имя элемента, к которому относятся атрибуты. Затем идет имя атрибута, его тип или список значений, которые он может принимать (все значения перечисляются через вертикальную черту, в скобках), признак обязательности присутствия атрибута в элементе или значение по умолчанию (это значение будет использовано, если атрибут не записан явно в XML документе).

Тип атрибута записывается одним из ключевых слов:

* CDATA – строка символов.
* ID – уникальный идентификатор, однозначно определяющий элемент, в котором встретился этот атрибут; значения такого атрибута не должны повторяться в документе. Используется для БД.
* IDREF – идентификатор, содержащий одно из значений атрибутов типа id, используется в качестве ссылки на другие элементы.
* IDREFS – идентификатор, содержащий набор значений атрибутов типа id, перечисленных через пробелы; тоже используется в качестве ссылки сразу на несколько элементов.
* ENTITY – имя сущности, объявленной в этом же описании DTD.
* ENTITIES – имена нескольких сущностей.
* NMTOKEN – слово, содержащее только символы, применяемые в именах. Атрибуты этого типа могут содержать имена других элементов или атрибутов, например, для того чтобы ссылаться на них.
* NMTOKENS – несколько таких слов, перечисленных через пробелы.
* NOTATION – обозначение, расшифрованное в описании DTD.

Признак обязательности записывается с использование ключевых слов:

#REQUIRED – атрибут надо обязательно записывать в элементе;

#IMPLIED – атрибут необязателен;

#FIXED – у атрибута возможно только одно значение, которое записывается тут же, через пробел.

Примеры:

<!ATTLIST city type (город | поселок | деревня) "город">

<!ATTLIST city name CDATA #REQUIRED>

<!ATTLIST pre xml:lang NMTOKEN "ru\_RU">

<!ATTLIST img

src CDATA #REQUIRED

id ID #IMPLIED

sort CDATA #FIXED "true"

print (yes | no) "yes"

>

## Объявление сущности

Ссылки на сущности используются как краткие обозначения для громоздких или часто повторяющихся фрагментов документа XML. Сами сущности, подставляемые в документ вместо ссылок, объявляются в описании DTD.

Все сущности можно разделить на три группы:

1. внутренние сущности – задаются при объявлении сущности. Объявление начинается с символов <!ENTITY, после которых через пробел записывается имя сущности и ее значение в кавычках. Например:

<!ENTITY lang "XML">

После такого объявления программа-анализатор, увидев в документе ссылку на сущность &lang;, заменить ее на строку XML. Ссылку на сущность можно применять тут же, в описании DTD, уже в следующем объявлении.

1. внешние сущности – содержатся в отдельных файлах или встроены в программу-анализатор. Для них указывается одно из слов SYSTEM или PUBLIC после которого записывается место их расположения (как для самого DTD).
2. параметризованные сущности – используются только внутри описания DTD. Объявление начинается с символов <!ENTITY, после которых через пробел записывается знак процента (%), имя сущности и ее значение в кавычках. Например:

<!ENTITY % lang "ru\_RU">

Ссылка на параметризованную сущность начинается не с амперсанда, а со знака процента, в примере %lang;. Введение этой ссылки удобно тем, что при смене языка надо будет поменять значение ru\_Ru только в одном месте описания.

## Объявление инструкций по обработке

Объявление инструкций по обработке начинается с символов <!NOTATION, после них записывается имя инструкции, одно из ключевых слов SYSTEM или PUBLIC (причем слова SYSTEM и PUBLIC здесь равнозначны), затем в кавычках – ее расшифровка.

Пример.

<!NOTATION image-gif SYSTEM "viewer.ехе">

Это объявление связывает обозначение image-gif с программой обработки изображений, находящейся в файле viewer.exe.

## Другие объявления

В DTD, как и в XML, можно (и нужно!) использовать комментарии.

<!-- комментарий -->

Отдельные фрагменты DTD можно временно включать/исключать из объявления. Для этого используются директивы IGNORE и INCLUDE.

<![IGNORE[

<!-- этот фрагмент исключен из обработки -->

<!ATTLIST BOOK Category CDATA "fiction">

<![INCLUDE[

<!-- кроме этой части -->

<!ELEMENT TITLE (#PCDATA)>

]]>

<!ELEMENT AUTHOR (#PCDATA)>

]]>

Вместо IGNORE можно просто закомментировать часть объявлений, но тогда воспользоваться INCLUDE будет нельзя.

## Недостатки и особенности DTD

Описание структуры документа XML, выполненное средствами DTD имеет ряд недостатков и ограничений:

1. Синтаксис DTD отличается от синтаксиса XML. Если мы хотим программно обрабатывать определения DTD, нам потребуется отдельный механизм анализа.
2. Определения DTD представляют собой закрытые конструкции. Вся информация должна быть включена только в одно определение DTD. Это не мешает, пока нет необходимости позаимствовать декларацию или какую-либо иную полезную конструкцию из другого определения DTD.
3. Сегментация недопустима, если не считать ситуаций, когда определение DTD можно логически сконструировать таким образом, чтобы оно включало вложенные определения DTD.
4. В DTD практически отсутствует информация о типах данных, точном количестве повторений вложенного элемента и некоторые другие необходимые подробности.

Необходимо отметить, что в настоящее время существует тенденция отказа от DTD в пользу XML Schema, поскольку средств DTD становится недостаточно.