Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» имени В.И. Ульянова (Ленина)

Кафедра Автоматики и Процессов Управления

**РЕФЕРАТ**

НА ТЕМУ

«DTD – DOCUMENT TYPE DEFINITION»

**Выполнил:** студент группы 3371

Потапова В.Э.

**Преподаватель:** Кондратьева Н.Ю.

Санкт-Петербург

2015

**Содержание:**

1. **Введение……………………………………………………………………………………………………….3**
2. **Объявление DTD в XML документе……………………………………………………………..4**
3. **Внутренние и внешние DTD………………………………………………………………………….4**
4. **Элементы**

**4.1 Типы элементов………………………………………………………………………………….6**

**4.2 Объявление элементов………………………………………………………………………6**

1. **Атрибуты**

**5.1 Общие понятия……………………………………………………………………………………8**

**5.2 Типы атрибутов……………………………………………………………………………………9**

**5.3 Типы значений атрибутов…………………………………………………………………..9**

**5.4 Определение атрибутов…………………………………………………………………….10**

**5.5 Пример описания атрибутов…………………………………………………………….11**

1. **Объекты**

**6.1 Общие понятия………………………………………………………………………………….12**

**6.2 Текстовый объект………………………………………………………………………………12**

* 1. **Двоичный объект………………………………………………………………………………12**
  2. **Параметрический объект………………………………………………………………….13**

**6.5 Символьные и числовые объекты…………………………………………………….14**

**7. Использованная литература………………………………………………………………………..15**

**Введение**

Спецификация [SGML](http://ru.wikipedia.org/wiki/SGML) предусматривает создание новых языков разметки.  
[HTML](http://www.w3.org/html/) и [XML](http://secretsofinternet.ru/2009/02/20/dtd-first-look/www.w3.org/XML) как раз таковыми и являются. XML документы могут быть двух типов:

* корректные (well-formed), удовлетворяющими определенному набору требований ;
* действительные (valid), т.е. соответствующие всем требованиям, предъявляемым к корректным XML документам, и удовлетворяющие условиям, подключенной к ним схемы.

С проверкой документа на корректность проблем не возникает: если ошибок не выскочило и всё отобразилось так, как мы хотели, то документ корректен. Например, если в HTML-документе написать что-то вроде «<Z>Привет!</Z>», то наш документ будет полностью корректен, но проигнорирован браузером. Почему? Потому что браузер ничего не знает о том, что это за «Z» такой. И если мы проверим наш документ на допустимость с помощью валидатора (валидатор - это программное обеспечение, выполняющее проверку кода на соответствие общепринятым стандартам консорциума), то документ таковым признан не будет. А как об этом узнает валидатор и на основании чего он вынес такой вердикт?

В настоящее время World Wide Web Consortium (W3C) поддерживает две схемы: DTD (Document Type Definition) и XML Schema.

Допустимость проверяется с помощью определения типа документа (DTD, document type definition). DTD может быть описан как внутри документа, так и вынесен в отдельный файл (аналогия с CSS: встроенные и подключаемые таблицы стилей). Описание DTD выполнено в Рекомендации «Extensible Markup Language (XML) 1.0», т.е. в той же самой рекомендации, где описаны требования к XML документу.

Задача DTD определить:

* состав элементов, которые могут использоваться в XML документе;
* описание моделей содержания, т.е. правил вхождения одних элементов в другие;
* состав атрибутов, с какими элементами XML документа они могут использоваться;
* каким образом атрибуты могут применяться в элементах;
* описание сущностей, включаемых в XML документ.

**Объявление DTD в XML документе**

Прежде чем начинать изучение правил составления DTD рассмотрим, как XML документ объявляет DTD.

DTD, относительно XML документа, могут быть двух типов: внешними и внутренними. Также возможно в XML документе одновременно использовать оба типа DTD.

Для объявления DTD в XML документе используется предложение **DOCTYPE**, которое должно располагаться перед корневым элементом XML документа, а указываемое в нем имя  DTD должно совпадать с именем корневого элемента. Предложение DOCTYPE имеет вид:

***<!DOCTYPE имя\_dtd  расположение\_dtd >*,**

где: ***имя\_dtd***  – имя схемы DTD, совпадающее с именем корневого элемента XML документа; ***расположение\_dtd*** – описатель местоположения схемы DTD, вид которого зависит от типа DTD.

**Внутренние и внешние DTD**

**Внешний DTD** находится в отдельном файле, который должен иметь расширение .dtd, например, library.dtd. Синтаксис предложения DOCTYPE в случае внешнего подключения принимает следующий вид:

***<!DOCTYPE имя\_dtd  http://genberm.narod.ru/xml/lections/dtd/dtd.files/image002.gif >***

где: ***URI*** – указание местоположения схемы DTD. ***SYSTEM и PUBLIC*** – альтернативно используемые ключевые слова, применяемые по следующему алгоритму.

При указании ***SYSTEM*** анализатор должен найти ***DTD*** по явно указанному ***URI***, например:

***<!DOCTYPE  library SYSTEM “http://www.library.org/dtd/library.dtd”>***

При указании ***PUBLIC*** считается, что будет использоваться публичный *DTD*, «хорошо знакомый» используемому обработчику XML документа. Предположим, что для библиотечных систем существует публичная схема DTD, расположенная на определенном сайте в Интернет. Если для работы с библиотечными XML документами используется не стандартный, а специализированный анализатор, то он может сам знать, где находится необходимая ему схема DTD. В этом случае не обязательно указывать точное URI расположения файла со схемой DTD, например:

***<!DOCTYPE  library PUBLIC “world/library.dtd”>***

Можно совместить описание ***РUBLIC***с указанием конкретного расположения файла со схемой DTD. В этом случае, если анализатор не найдет публичного схему, он будет использовать схему, указанную по конкретному адресу, например:

***<!DOCTYPE  library PUBLIC "world/library.dtd"***

***"http://www.library.org/dtd/library.dtd">***

**Внутренние DTD** располагаются в том же файле, что и сам XML документ.

Внутренние DTD описываются предложением DOCTYPE, которое имеет следующий вид:

***<!DOCTYPE library [***

***…***

***]>***

*Правила DTD располагаются между двумя квадратными скобками.* Пример XML документа с внутренним DTD приведен ниже:

***<?xml library version=”1.0” encoding=”Windows-1251”?>***

***<!DOCTYPE library [ …***

***>]***

***<library>***

***<book>***

***<title> … </title>***

***<author> … </author>***

***</book>***

***</library>***

**Одновременное использование внешнего и внутреннего DTD**

При использовании внешнего DTD может возникнуть ситуация, когда какие-либо его описания могут не удовлетворять. Например, в XML документе необходимо использовать несколько новых элементов или изменить тип ряда атрибутов. Непосредственно корректировать внешний DTD достаточно нельзя, так как его могут использовать другие программы, и можно нарушить правильность их работы. В такой ситуации рекомендуется использовать внешний DTD, а необходимые изменения выполнить во внутреннем DTD.

*Примечание*. В случае возникновения конфликта между описаниями внешнего и внутреннего DTD приоритет имеют описания внутреннего DTD. Синтаксис предложения DOCTYPE при одновременном использовании внешнего и внутреннего DTD имеет следующий вид:

***<!-- сначала внешний, а за ним внутренний DTD -->***

***<!DOCTYPE имя\_dtd  http://genberm.narod.ru/xml/lections/dtd/dtd.files/image006.gif >***

***[ содержание внутреннего DTD***

***]>***

***<!-- XML документ -->***

***<library>***

***. . .***

***</library>***

***>***

**Элементы**

**Типы элементов**

XML документ состоит из набора тегов. С точки зрения DTD все теги, представленные в XML документе являются элементами и должны быть соответствующим образом описаны.

Теги XML документа могут быть двух типов:

* *контейнерные*, представляются парой тегов (начальный и конечный);
* *пустые*, представляются одиночным тегом. Признаком пустого тега является слеш перед закрывающейся угловой скобкой.

Контейнерный тег хранит в себе текст или другие теги. В примере, описывающем библиотечный каталог, все теги контейнерные. Так тег <book> является контейнерным, поскольку хранит в себе другие теги <title> и <author>. Соответственно теги <title> и <author> являются контейнерными, поскольку хранят в себе текст: название книги и имя автора.

Пустые теги не могут содержать в себе текст или другие теги. Обычно они используются для включения объектов. Что такое объекты, и какие виды объектов существуют, описывается в разделе «Объекты» данного документа. Здесь, в качестве примера, покажем, как можно включить в текст XML документа такой объект, как графический файл:

*<image file="c:\image\image.gif" />*

**Объявление элементов**

Для объявления элементов в DTD используется предложение ELEMENT, которое имеет следующий вид:

***<!ELEMENT имя\_элемента (модель\_содержания) [EMPTY]>***

где: ***имя\_элемента*** – имя, под которым элемент может появляться в XML документе;

***модель\_содержания*** – описание правил вхождения одних элементов в другие;

*EMPTY* – необязательный параметр, применяемый для указания о том, что элемент должен быть пустым.

В DTD должны быть объявлены все элементы, которые можно будет использовать в XML документе. Также для каждого элемента должна быть определена модель содержания, позволяющая определить:

* может ли данный элемент входить в другие элементы;
* какие элементы, в каком порядке и количестве могут входить в данный элемент.

При построении модели содержания необходимо помнить, что описание вложенности выполняется только для  непосредственно нижележащего уровня. Так элемент, формирующий контейнер первого уровня, может описывать структуру не всех входящих в него элементов и контейнеров, а только элементов следующего за ним второго уровня. Элементы второго уровня описывают структуру элементов и контейнеров третьего уровня и т.д.

В модель содержания могут включаться элементы одного из четырех типов:

* элементный;
* текстовый;
* смешанный;
* произвольный.

Такая классификация типов содержания элементов обусловлена, в первую очередь, возможностью включения в элемент текстовой информации.  Рассмотрим каждый из этих типов.

**Элементный тип** – состоит только из элементов и не может включать текст. В  представленном в начале лекции XML документе к данному типу относятся такие элементы как library, books,book.

Модель содержания для элементного типа позволяет не только указать, какие элементы будут содержаться в элементе-контейнере, но задать их порядок и количество вложений (ни разу, один раз, много раз). Для задания множественности вложений используется набор специальных символов:

* "," (запятая) – разделяет типы элементов при их перечислении в модели содержания;
* "|" (вертикальная черта) – разделяет альтернативные элементы при их перечислении (обычно в элементе смешанного типа);
* "+" (знак плюс) – определяет, что элемент будет использоваться один или несколько раз;
* "?" (вопросительный знак) – определяет, что элемент не будет использоваться вообще или будет использоваться только один раз;
* "\*" (звездочка) – определяет, что элемент не будет использоваться вообще или может использоваться несколько раз;
* "()" (круглые скобки) – обрамляют или выделяют последовательность элементов.

Пример описания элемента элементного типа:

***<!ELEMENT books (book+)>***

***<!ELEMENT book (udk, authors, publishing, annotation?)>***

***<!ELEMENT authors (author+)>***

В данном примере полагаем, что у книг, связанных с информационными технологиями, должен быть хотя бы один автор, а возможно и несколько авторов. Однако если бы разрабатывался DTD для библиотеки с художественными книгами, то  у книг, например, с народными сказками возможна ситуация, когда авторы не известны. В этом случае элемент book более правильно было бы объявить так:

***<!ELEMENT book (udk, authors?, publishing, annotation?)>***

**Текстовый тип** включает только текст. В представленном примере к данному типу относятся такие элементы как author, title, publishing. Данный тип идентифицируется словом **#PCDATA**(parsed character data) в поле модели содержания. Пример описания элемента текстового типа:

***<!ELEMENT author (#PCDATA)>***

**Смешанный тип** может включать как текст, так и элементы. В представленном примере к данному типу относится элемент annotation. Пример описания элемента смешанного типа:

***<!ELEMENT annotation (#PCDATA | scope)\*>***

На использование смешанного типа накладывается ряд ограничений:

* нельзя определить порядок следования элементов, входящих в смешанный тип;
* нельзя определить количество повторов элементов, входящих в смешанный тип;
* одно и то же имя не может быть представлено в объявлении смешанного типа более одного раза.

**Произвольный тип** может включать любое содержание. Характеризуется словом ANY в поле модели содержания. Пример описания элемента произвольного типа:

***<!ELEMENT имя\_элемента ANY>***

Создавать и использовать элементы произвольного типа обычно не рекомендуется.

**Атрибуты**

**Общие понятия**

Элементы описывают основные понятия предметной области. Атрибут уточняет информацию об этом понятии. Он позволяет понять, какая информация связана с элементами и, самое главное, контролировать тип этой информации. Например, возьмем элемент IMG из языка HTML, который предназначен для задания вывода изображения браузером на экран. (В первой лекции мы говорили, что HTML – это подмножество XML.)

<img src="http://www.mysite.com/image.gif" align= "left"                   alt="Пример изображения" border= "1" width= "150" height= "100" />

В примере элемент IMG имеет шесть атрибутов, которые уточняют различные понятия, связанные с выводом изображения.

Следует иметь в виду, что то же самое можно описать и без использования атрибутов, с помощью одних элементов.

<img>

          <src> http://www.mysite.com/image.gif</src>

          <align>left</align>

          <alt> Пример изображения</alt>

          <border>1</border>

          <width>150</width>

          <height>100</height>

</img>

Преимущество использования атрибутов состоит в том, как мы увидим далее, что они позволяют контролировать свои значения, чего нельзя достичь, используя только элементы.

**Типы атрибутов**

Существует три типа атрибутов:

* обязательные;
* фиксированные;
* неявные.

**Обязательные атрибуты** (уже видно из их названия) – это атрибуты, которые всегда должны быть определены для данного типа элемента. В противном случае анализатор будет считать документ не состоятельным и выдаст ошибку во время его синтаксического анализа.

**Фиксированные атрибуты** – это такие атрибуты значения которых разработчик XML документа не может изменять. Для таких атрибутов можно использовать либо конкретно указанное значение, либо одно из возможных значений. Фиксированные атрибуты могут иметь значение по умолчанию. В этом случае их в тегах XML документа указывать не обязательно, автоматически будут приняты значения по умолчанию.

**Неявные атрибуты** – это не обязательные и не фиксированные, т.е. все атрибуты, которые не подходят под определение обязательных и фиксированных атрибутов. Неявные атрибуты не имеют значения по умолчанию и не обязаны всегда появляться вместе с элементом к которому они относятся.

**Определение типов атрибутов.** В DTD типы атрибутов задаются специальными идентификаторами, значение которых представлены в таблице 1.

|  |  |
| --- | --- |
| Тип атрибута | Идентификатор |
| Обязательный | #REQUIRED |
| Фиксированный | #FIXED |
| Неявный | #IMPLIED |
|  |  |

**Типы значений атрибутов**

Атрибуты могут принимать следующие значения:

* простой текст;
* уникальный идентификатор;
* предопределенное значение;
* объект.

**Простой текст** – это строки текста, задаваемые разработчиком. Значения неявных атрибутов, как правило, именно простой текст. Например, мы хотим для книг библиотечного каталога ввести атрибут – адрес сайта издательства в Интернет.

***<book site= www.booksite.com>***

Тип значения атрибута – простой текст.

**Уникальный идентификатор** – это текстовая строка, уникально идентифицирующая определенный элемент, которая имеет следующий формат:

***ID= "уникальный\_идентификатор"***

Обычно уникальный идентификатор используется в том случае, когда применяются таблицы стилей, чтобы связать элемент с определенным стилем. В XML нет никаких правил, как назначать значения уникальному идентификатору. Единственное ограничение – должен начинаться с алфавитно-цифрового символа.

**Предопределенное значение** – это множество заранее определенных значений только одно, из которых может выбрать разработчик XML документа для присвоения атрибуту. Одно из значений множества может использоваться как значение по умолчанию. Например, в HTML атрибут ALIGN, отвечающий за расположение текста на строке может принимать только одно из трех предопределенных ему значений {LEFT, CENTER, RIGHT}.

Достоинство использования предопределенных значений атрибута заключается в том, что они дают дополнительный уровень контроля, гарантируя установку правильное значение.

**Объект** – это виртуальный блок памяти, который может хранить определенную информацию. До того, как объект будет включен в XML документ, он должен быть объявлен, используя определенное объявление. Более подробно типы объектов и их объявления будут описаны далее в разделе "Объекты".

**Определение атрибутов**

Синтаксис объявления атрибутов имеет следующий формат:

***<!ATTLIST имя\_элемента имя\_атрибута***

***тип\_значения тип\_атрибута "значение\_по\_умолчанию"***

С одним элементом может связано несколько атрибутов, поэтому все параметры, кроме имя\_элемента, могут повторяться. В предложении ATTLIST тип значения атрибута представлен идентификаторами, значения которых даны в таблице 2.

|  |  |
| --- | --- |
| Тип | Иденификатор |
| Простой текст | CDATA |
| Уникальный идентификатор | ID |
| Предопределенное значение | (значение | значение |…|значение |
| Нетекстовый объект | ENTITY |
|  |  |

**Пример описания атрибутов**

Допустим, в используемом нами XML документе, есть элемент image, атрибуты которого определяют графический файл и правила его расположения на экране:

***<image id="PIC1" border="0" alt="просто рисунок"***

***src="www.mysite.com/image/image.gif" align="center" />***

 Описание элемента image для нашего примера будет вид:

|  |  |  |
| --- | --- | --- |
| ***<!ELEMENT image empty>*** | | ***1*** |
| ***<!ATTLIST image id ID #REQUIRED*** | | ***2*** |
|  | ***border CDATA #IMPLIED*** | ***3*** |
|  | ***alt CDATA #IMPLIED*** | ***4*** |
|  | ***src ENTITY #REQUIRED*** | ***5*** |
|  | ***align (left,center,right) #REQUIRED "left"*** | ***6*** |
|  | ***valign CDATA #FIXED "middle"*** | ***7*** |

*Примечание.* Цифры 1-7 в правой части примера служат только для идентификации строк и не являются частью примера.

*Строка 1* описывает элемент image. Параметр empty показывает, что элемент image пустой.

*Строка 2* задает список атрибутов и описывает уникальный идентификатор ID, который в данном примере является обязательным.

*Строка 3* описывает атрибут border как неявный, простой текст.

*Строка 4* описывает атрибут alt как обязательный, простой текст.

*Строка 5* описывает атрибут src как обязательный, нетекстовый объект.

*Строка 6* описывает атрибут align как обязательный, предопределенное значение, значение по умолчанию lefr.

*Строка 7* описывает атрибут valign как фиксированный, простой текст, значение по умолчанию middle. В примере этого атрибута нет, но поскольку он фиксированный, то используется неявно, и представляется своим значением по умолчанию.

**Объекты**

**Общие понятия**

**Объект** – это виртуальный блок памяти, который может хранить определенную информацию. Тип информации, которую может хранить объект:

* текстовая;
* двоичная;
* параметрическая;
* символы не входящие в набор ASCII.

Синтаксис описания объекта имеет следующий вид:

***<ENTITY имя "содержание">***

где ***имя*** – это имя, которое будет иметь объект;

***содержание*** – значение объекта, связанное с именем.

*Значение параметра "содержание" зависит от типа объекта.*

**Текстовый объект**

**Текстовый объект** - это блок текста, который может часто появляться в тексте документа. В текстовый объект может и входить и фрагмент разметки. Например, в XML документе библиотечного каталога элемент author имеет обязательный атрибут country, который указывает на принадлежность автора к определенной стране. Для того, чтобы сократить запись можно использовать текстовый объект.

***<!ENTITY auth\_r "author country="Russian" ">***

Текстовый объект может быть внутренним, как в приведенном выше примере, и внешним, например:

***<!ENTITY auth\_r "c:\xml\author\russia.xml" SYSTEM>***

Вместо параметра SYSTEM может использоваться параметр PUBLIC. Правила использования параметров SYSTEM и PUBLIC полностью аналогичны правилам их использования в предложении DOCTYPE, которые были описаны ранее, в разделе "Внешние и внутренние DTD".

**Двоичный объект**

**Двоичный объект** – это обычно внешний объект, который определяет двоичный файл (графический файл, файл мультимедиа, исполняемую программу и т.п.). Признаком двоичного объекта является идентификатор NDATA. Синтаксис объявления двоичного объекта имеет следующий вид:

***<!ENTITY имя "URL" SYSTEM NDATA тип\_файла>***

где: ***URL*** определяет местонахождение двоичного объекта;

***тип\_файла*** – уточняет тип двоичного файла.

Например,

***<!ENTITY logo "www.mysite.com/logo/logo.gif"***

***SYSTEM NDATA gif>***

Поскольку анализатор может не знать, каким образом или с помощью какой программы обрабатывать двоичный объект, то ему необходимо подсказать. Для этой цели используется предложение NOTATION. Например, если для обработки gif файлов необходимо использовать программу PHOTOSHOP, то предложение NOTATION примет следующий вид:

***<!NOTATION gif SYSTEM "c:\apps\graphics\photoshop.exe">***

В том случае, когда анализатор знает как обрабатывать двоичный объект и имеет для этого собственные механизмы обработки, предложение NOTATION задавать не надо.

**Параметрический объект**

***Параметрический объект*** – предназначен для хранения списков атрибутов и моделей содержания. Если несколько разных элементов имеют одинаковые списки атрибутов, то для того, чтобы каждый раз не задавать их, можно воспользоваться параметрическим объектом. Синтаксис объявления параметрического объекта имеет следующий вид:

***<!ENTITY % имя "содержание">***

Например, мы хотим ввести в XML документ, описывающий библиотечный каталог, для каждой книги (элемент book) наименование издательства и картинку с логотипом этого издательства.

***<book>***

***<publish>… </publish>***

***<logo src="www.publish.com/logo/logo.gif" border="0"***

***alt="логотип издательства" align="center" />***

***</book>***

Поскольку значения всех атрибутов элементе logo кроме src будут повторяться для всех элементов book, то для упрощения записи можно задать параметрический объект pub\_logo:

***<!ENTITY % pub\_logo "***

***border CDATA #IMPLIED***

***alt CDATA #IMPLIED***

***align (left,center,right) #REQUIRED "left"***

***valign CDATA #FIXED "middle"***

Применив pub\_logo вид элемента logo будет иметь вид:

***<logo src="www.publish.com/logo/logo.gif" pub\_logo />***

**Символьные и числовые объекты**

**Символьный объект** – это комбинация знаков, которая должна представлять символы, не входящие в набор ASCII символов. По умолчанию XML использует кодировку ISO-Latin-1, набор символов которой шире, чем в набор ASCII символов. XML может использовать и другие кодировки. Тип кодировки указывается в командной инструкции xml, с которой всегда начинается XML документ:

***<?xml version="1.0" encoding="имя кодировки"?>***

Например, для того, чтобы использовать кириллицу необходимо указать кодировку Windows-1251:

***<?xml version="1.0" encoding=" Windows-1251"?>***

Кроме символов, не входящих в набор ASCII, символьные объекты используются для ряда символов, входящих в набор ASCII, но зарезервированных для специальных целей. Например,

* амперсант "&";
* знак меньше (открывающаяся угловая скобка) "<";
* знак больше (закрывающаяся угловая скобка) "<".

**Числовой объект** – это комбинация цифр, которая служит для тех же целей, что и символьные объекты, и может использоваться как альтернативный способ описания.

В таблице 3 представлено несколько символьных и соответствующих им числовых объектов.

|  |  |  |
| --- | --- | --- |
| Специальные символы | Символьный объект | Числовой объект |
| "&" | &amp | &#38 |
| "<" | &lt | &#60 |
| ">" | &gt | &#62 |

**Использованная литература:**

[**http://gberman.narod.ru/xmllections/dtd/dtd/dtd.htm**](http://gberman.narod.ru/xmllections/dtd/dtd/dtd.htm)

[**http://genberm.narod.ru/xml/lections/dtd/dtd.html**](http://genberm.narod.ru/xml/lections/dtd/dtd.html)