# Язык разметки XML

# Общие сведения об XML

**XML (eXtensible** **Markup** **Language)** – рекомендованный W3C язык разметки. XML – текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе специализированных языков разметки. XML является упрощенным подмножеством языка SGML .

В 1986 году, задолго до того, как идея создания сети Интернет была воплощена в жизнь, универсальный стандартизированный язык разметки SGML (Standardized Generalized Markup Language) был утвержден в качестве международного стандарта (ISO 8879) определения языков разметки.

Язык HTML первоначально был всего лишь одним из SGML-приложений. Он описывал правила, по которым должна быть подготовлена информация для World Wide Web. Таким образом, язык HTML – это набор предписаний SGML, сформулированных в виде определения типа документа (DTD), объясняющих, что именно обозначают тэги и элементы. Схема DTD для языка HTML хранится в веб-браузере.

К недостаткам языка HTML можно отнести следующие:

* HTML имеет фиксированный набор тэгов. Нельзя создавать свои тэги, понятные другим пользователям.
* HTML – это исключительно технология представления данных. HTML не несет информации о значении содержания, заключенного в тэгах.
* HTML – "плоский" язык. Значимость тэгов в нем не определена, поэтому с его помощью нельзя описать иерархию данных.
* В качестве платформы для приложений используются браузеры. HTML не обладает достаточной мощью для создания веб-приложений на том уровне, к которому в настоящее время стремятся веб-разработчики. Например, на языке HTML невозможно разработать приложение для профессиональной обработки и поиска документов.
* Большие объемы трафика сети. Существующие HTML-документы, используемые как приложения, перегружают Интернет большими объемами трафика в системах клиент-сервер. Примером может служить пересылка по сети большого по объему документа, в то время как необходима только небольшая часть этого документа.

Таким образом, с одной стороны, язык HTML является очень удобным средством разметки документов для использования в WWW, а с другой – документ, размеченный в HTML, имеет мало информации о своем содержании.

Если тот или иной документ несет достаточно полную информацию о своем содержании, появляется возможность сравнительно легко провести автоматическую обобщенную обработку и поиск в файле, хранящем документ. Язык SGML позволяет сохранять информацию о содержании документа, однако вследствие особой сложности он никогда не использовался так широко, как HTML.

Группа экспертов по языку SGML, возглавляемая Джоном Боузэком (Jon Bosak) из компании Sun Microsystems, приступила к работе по созданию подмножества языка SGML, которое могло бы быть принято Web-сообществом. Решено было удалить многие несущественные возможности SGML. Перестроенный таким образом язык назвали XML. Упрощенный вариант оказался значительно более доступным, чем оригинал, его спецификации занимали всего 26 страниц по сравнению с более чем 500 страницами спецификаций SGML.

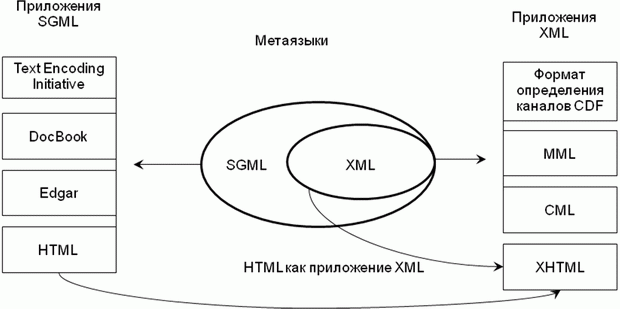
Язык XML имеет следующие достоинства:

* Это человеко-ориентированный формат документа, он понятен как человеку, так и компьютеру.
* Поддерживает Юникод.
* В формате XML могут быть описаны основные структуры данных - такие как записи, списки и деревья.
* Это самодокументируемый формат, который описывает структуру и имена полей также как и значения полей.
* Имеет строго определенный синтаксис и требования к анализу, что позволяет ему оставаться простым, эффективным и непротиворечивым.
* Широко используется для хранения и обработки документов;
* Это формат, основанный на международных стандартах;
* Иерархическая структура XML подходит для описания практически любых типов документов;
* Представляет собой простой текст, свободный от лицензирования и каких-либо ограничений;
* Не зависит от платформы;
* Является подмножеством SGML, для которого накоплен большой опыт работы и созданы специализированные приложения;

К известным недостаткам языка можно отнести следующие:

* Синтаксис XML избыточен.
  + Размер XML документа существенно больше бинарного представления тех же данных (порядка 10 раз).
  + Размер XML документа существенно больше, чем документа в альтернативных текстовых форматах передачи данных (например JSON, YAML) и особенно в форматах данных, оптимизированных для конкретного случая использования.
  + Избыточность XML может повлиять на эффективность приложения. Возрастает стоимость хранения, обработки и передачи данных.
  + Для большого количества задач не нужна вся мощь синтаксиса XML, и можно использовать значительно более простые и производительные решения.
* Пространства имен XML сложно использовать и их сложно реализовывать в анализаторах.
* XML не содержит встроенной в язык поддержки типов данных. В нем нет понятий "целых чисел", "строк", "дат", "булевых значений" и т.д.
* Иерархическая модель данных, предлагаемая XML, ограничена по сравнению с реляционной моделью и объектно-ориентированными графами.

По сути, XML служит метаязыком для описания структуры других языков. Взаимосвязь между SGML, XML, HTML и некоторыми другими языками показана на следующей диаграмме:



Важным отличием XML от HTML является то внимание, которое уделяется контролю за тем, насколько точно соблюдаются правила языка при разметке документов. В зависимости от этого принято выделять *правильно построенные* и *действительные* XML документы .

Документ XML считается **правильно построенным (well-formed)**, если он соответствует всем синтаксическим правилам XML:

1. XML документ содержит один и только один корневой элемент, содержащий все остальные элементы
2. Дочерние элементы, содержащиеся в корневом элементе, должны быть правильно вложены.
3. Имена элементов подчиняются правилам:
   * имя начинается с буквы, знака подчеркивания или двоеточия;
   * после первого символа в имени могут быть буквы, цифры, знаки переноса, подчеркивания, точка или двоеточие;
   * имена не могут начинаться с буквосочетания XML.

В отличие от HTML, в XML учитывается регистр символов, т.е. book и Book – это разные имена. Последовательные пробелы не сокращаются.

XML документ имеет следующую структуру.

Первая строка XML документа называется **объявлением XML**. Это необязательная строка, указывающая версию стандарта XML (обычно это 1.0). Также здесь может быть указана кодировка символов и внешние зависимости.

**Комментарий** может быть размещен в любом месте документа. XML комментарии размещаются внутри пары тегов <!-- и заканчиваются -->. Два знака дефис (--) не могут быть применены ни в какой части внутри комментария.

Остальная часть XML-документа состоит из вложенных **элементов**, каждый из которых обрамляется парой открывающего и закрывающего тегов. Содержимым элемента называется все, что расположено между открывающим и закрывающим тегами, включая текст и другие (вложенные) элементы. Кроме содержания у элемента могут быть атрибуты - пары имя=значение, добавляемые внутрь открывающего тега после названия элемента.

<элемент атрибут1="значение" атрибут2="значение">содержимое элемента</элемент>

Пропускать закрывающий тег нельзя. Но если у элемента нет содержимого, его можно сократить (пустой элемент):

<элемент атрибут1="значение" атрибут2="значение" />

Значения атрибутов всегда заключаются в кавычки (одинарные или двойные), одно и то же имя атрибута не может встречаться дважды в одном элементе. Не рекомендуется использовать разные типы кавычек для значений атрибутов одного тега.

атрибут1="значение"

атрибут1=’значение’

Более подробно синтаксис XML рассматривается ниже.

К сожалению, описанные правила позволяют контролировать только формальную правильность XML документа, но не содержательную.

Проверка **действительности (valid)** документа предполагает выполнение следующих действий:

1. Проверка использования только разрешенного набора элементов.
2. Проверка полного соответствия порядка следования элементов и атрибутов содержанию документа или определенным правилам.
3. Контроль типов данных.
4. Контроль целостности данных.

Таким образом, действительный документ дополнительно соответствует некоторым семантическим правилам. Это более строгая дополнительная проверка корректности документа на соответствие заранее определённым, но уже внешним правилам, в целях минимизации количества ошибок.

Эти правила могут быть разработаны как самим пользователем, так и сторонними разработчиками. Обычно такие правила хранятся в специальных файлах – схемах, где самым подробным образом описана структура документа, все допустимые названия элементов, атрибутов и многое другое.

**Схема** четко определяет имя и структуру корневого элемента, включая спецификацию всех его дочерних элементов. Программист может задать, какие элементы и в каком количестве обязательны, а какие – необязательны. Схема также определяет, какие элементы содержат атрибуты, допустимые значения этих атрибутов, в т.ч. значения по умолчанию.

Чаще всего для описания схемы используются следующие спецификации:

* DTD (Document Type Definition)
* XDR (XML Data Reduced)
* XSD (язык определения схем XML)

XML документ отличается от HTML документа также и тем, как он отображается в веб-браузере. Сам по себе XML-документ отображается как простой текст в большинстве веб-браузеров. Некоторые веб-браузеры, такие как Internet Explorer и Firefox, отображают структуру документа в виде дерева, позволяя сворачивать и разворачивать узлы с помощью нажатий клавиши мыши.

Наиболее распространены три способа преобразования XML-документа в отображаемый пользователю вид:

* применение стилей CSS;
* применение преобразования XSLT;
* написание на каком-либо языке программирования обработчика XML-документа.

# Информационная модель

Разработка формата XML начинается с создания информационной модели.

**Информационная модель –** это описание используемой в системе информации, не зависящее от какой бы то ни было информационной технологии. Информационная модель описывает назначение данных.

Существуют два основных типа информационной модели: статическая и динамическая. Разрабатываемая модель XML будет *статической*, поскольку она отражает структуру и объекты системы, но не отражает их поведение и действия с ними.

При построении статической информационной модели необходимо пройти следующие этапы:

Этап 1. Идентификация понятий, присвоение им имен и их определение

Этап 2. Организация понятий в иерархию классов (таксономия)

Этап 3. Определение связей, множественности и ограничений

Этап 4. Добавление свойств для конкретизации деталей значений, связанных с объектами

## Именование понятий

Для начала нужно составить список понятий, относящихся к системе, придумать для них имена, краткое описание.

Следует выбирать такие имена, чтобы они были краткими, но понятными. Описание должно быть точным, чтобы не возникало разногласий по существу определения.

В конце этого этапа мы получим список типов тех объектов с именами и определениями.

## Таксономия

*Таксономия* — это термин, используемый в биологии для обозначения системы классификации. В информационном моделировании ее также называют *иерархией типов*.

Перечислив и назвав типы объектов, их надо организовать в иерархическую схему классификации. Часто эти иерархические отношения возникают уже на этапе определения типов объектов.

Ключевой здесь является фраза, определяющая принадлежность (англ. «is a kind of», AKO). Написав предложение вида "А есть разновидность В" или "Каждое А есть В", вы определили отношения подтипов в вашей таксономии.

Этап 2 сводится к организации типов объектов в иерархию типов.

## Поиск связей

После того как объекты названы, в статическом информационном моделировании надо определить *связи*, существующие между ними. Связи можно показать просто сформулировав их в виде обычных предложений или их можно показать графически в виде диаграммы. Диаграммы следует делать предельно простыми и интуитивно понятными.

Множественность связи показывает, сколько объектов каждого типа принимает в ней участие.

* Связи типа "один-ко-многим": одна глава содержит много параграфов, один человек покупает много туристических поездок.
* Связи типа "многие-ко-многим" также часто встречаются: один автор может написать несколько книг, но у книги также может быть несколько авторов.
* Связи типа "один-к-одному".

При моделировании информации для окончательного представления XML особенно важным типом связей являются связи включения (англ. «a part of», APO). Множественность этих связей всегда бывает "один-ко-многим" и "один-к-одному".

Возможны две формы связей включения. Первая – это *агрегации*, относительно свободное объединение объектов, позволяющее рассматривать их группу в течение некоторого времени как целое (например, туристическая группа, одни и те же люди могут в разное время входить в разные группы). Вторая форма – *композиция*. Это более строгая форма. Отдельные части целого не могут существовать независимо от него (например, комнаты в отеле не могут существовать независимо от отеля).

Итак, мы определили связи, существующие между типами объектов в нашей модели.

## Описание свойств

Свойства представляют собой простые значения, ассоциированные с объектами. У человека можно определить рост, вес, национальность и род занятий; отель имеет определенное количество комнат, этажность и ценовую категорию.

Не следует снова включать связи в список свойств объекта: расположение отеля не является его свойством, если мы уже промоделировали его как связь с курортом.

Главное, что надо знать о свойствах, - это тип их данных. Определен ли для них фиксированный диапазон значений, являются ли они числовыми, в каких единицах выражаются? Является ли свойство обязательным и есть ли у него значение по умолчанию?

В конце этапа 4 мы завершили формирование статической информационной модели: получили полное описание типов объектов в системе, их связей друг с другом и их свойств.

# Синтаксис XML

Для ограничения тегов в разметке XML, так же как и в HTML, используются угловые скобки: тег начинается со знака "меньше" (<) и завершается знаком "больше" (>). Но необходимо помнить, что в отличие от HTML вся разметка XML чувствительна к регистру символов, это касается как имен тегов, так и значений атрибутов.

## 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.

## Пространства имен XML

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">

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.

В дальнейшем имена тегов и атрибутов, которые мы хотим отнести к пространству имен "http://URI\_namespace", снабжаются префиксом ns, например:

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

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.

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.

## Элементы

Документ XML состоит из элементов. Элемент начинается открывающим тегом, затем идет необязательное содержимое элемента, после чего записывается закрывающий тег. В отличие от HTML наличие закрывающего тега обязательно, исключением являются элементы без содержания, так называемые пустые элементы, которые могут быть записаны в сокращенной форме.

#### Открывающие теги

Открывающий тег начинается со знака "меньше" (<) и завершается знаком "больше" (>), внутри которых помещаются имя элемента:

<имя\_элемента>

#### Закрывающие теги

Закрывающий тег начинается со знака "меньше" (<) за которым следует "косая черта" (/) после которой повторяется имя элемента из соответствующего открывающего тега и завершается знаком "больше" (>):

</имя\_элемента>

При этом необходимо помнить, что каждый закрывающий тег должен соответствовать своему открывающему тегу, а так же что вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов.

Таким образом, полностью элемент выглядит следующим образом:

<имя\_элемента> содержание элемента </имя\_элемента>

#### Пустой элемент

Если элемент не имеет содержимого, его можно записать двумя способами:

<имя\_элемента></имя\_элемента>

<имя\_элемента />

В качестве содержимого элемента могут выступать

* другие элементы,
* символьные данные,
* ссылки на символы,
* ссылки на сущности,
* комментарии,
* разделы CDATA,
* инструкции по обработке.

#### Ссылки на символы

Для того чтобы вставить в текст документа некоторый символ, который, например не присутствует в раскладке клавиатуры либо может быть неправильно истолкован анализатором, используют ссылки на символы. Ссылка на символ обязательно начинается со знака "&" (амперсанта) и заканчивается точкой с запятой. Ссылки на символы записываются в следующем виде:

**&#**код\_символа\_в\_Unicode**;**

Код символа можно записать и в шестнадцатеричном виде. В этом случае перед ним ставится символ "x":

**&#x**Шестнадцатеричный\_код\_символа**;**

Кроме этого существуют именованные подстановки, определенные в спецификации XML, и реализованные во всех совместимых с XML анализаторах, которые делают текст документа более понятным для человека. С помощью этих именованных подстановок можно вставить в текст документа такие символы как:

|  |  |
| --- | --- |
| **Символы** | **Именованные подстановки** |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ' | &apos; |
| " | &quot; |

#### Ссылки на сущности

Ссылки на сущности позволяют включать любые строковые константы в содержание элементов или значение атрибутов. Ссылки на сущности, как и ссылки на символы, начинающиеся с амперсанта, после которого идет имя сущности и заканчивающиеся точкой с запятой:

**&**имя\_сущности**;**

Ссылки на сущности указывают программе-анализатору подставить вместо них строку символов, заранее заданную в определении типа документа (DTD).

#### Разделы CDATA

Секция CDATA используется, для того чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы. Программа-анализатор не разбивает секцию CDATA на элементы, а считает ее просто набором символов. В отличие от комментариев, содержание данной секции не игнорируется, а передается без изменений на выход программы анализатора, благодаря чему его можно использовать в приложении.

Секция CDATA начинается со строки <![CDATA[, после которой записывается содержимое секции. Завершается секция двумя закрывающими квадратными скобками и знаком "меньше":

**<![CDATA[** содержание секции **]]>**

Секция CDATA может содержать любую символьную строку, кроме "]]>".

Секции CDATA можно создать в содержимом любого элемента XML. На секцию налагаются следующие два ограничения:

1. внутри секции CDATA нельзя создать новую секцию,
2. секции CDATA нельзя вкладывать друг в друга.

#### Инструкции по обработке

Назначение инструкций по обработке — сообщить информацию, передаваемую XML-процессором приложению. Инструкция по обработке имеет следующую общую форму записи:

<? Кому инструкция ?>

Здесь *Кому* есть имя приложения, которому адресована инструкция. Допускается любое имя при соблюдении следующих правил:

* имя должно начинаться с буквы или символа подчеркивания (\_), после чего могут следовать или не следовать другие буквы, цифры, точки (.), тире (-) или символы подчеркивания (\_);
* имя «xml», в любом сочетании строчных или прописных букв, зарезервировано («xml» строчными буквами используется в объявлении XML-документа, которое представляет собой разновидность инструкции по обработке).

*Инструкция* есть информация, передаваемая приложению. Она может состоять из любой последовательности символов, за исключением пары ?>, зарезервированной для обозначения окончания инструкции по обработке.

Например, следующая инструкция по обработке предписывает браузеру использовать CSS-таблицу из файла styles.css:

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

#### Комментарии

Комментарии помещаются в специальные скобки и не анализируются. Они могут располагаться в любой части документа.

<!-- Текст комментария -->

Нельзя использовать двойной дефис (--) внутри комментария.

## Структура XML- документа

Любой XML-документ состоит из следующий частей:

* Необязательный пролог
* Тело документа (корневой элемент)
* Необязательный эпилог, следующий за деревом элементов

### Пролог

Пролог состоит из нескольких частей:

* объявление XML;
* объявление типа документа;
* комментарии;
* команды обработки;
* объявления пространств имен.

Поскольку все эти части необязательны, пролог может быть опущен.

#### Объявление XML (XML Declaration)

Объявление заключено между символами <?...?> и содержит:

* пометку xml и номер версии (*version*) спецификации XML;
* указание на кодировку символов (*encoding*), в которой написан документ (по умолчанию encoding="UTF-8");
* параметр standalone, который может принимать значения "yes" или "no" (по умолчанию standalone="yes"). Значение "yes" показывает, что в документе содержатся все требуемые декларации элементов, a "no" - что нужны внешние определения DTD.

Все это вместе может выглядеть следующим образом:

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

Важно отметить, что в объявлении XML только атрибут *version* является обязательным, все остальные атрибуты могут быть опущены и, следовательно, принимать значения по умолчанию. Также нужно помнить, что все эти атрибуты следует указывать только в приведенном порядке.

#### Объявление типа документа DTD (Document Type Declaration)

Записывается в виде <!DOCTYPE...> и может занимать несколько строк. В этой части объявляются теги, использованные в документе, или приводится ссылка на файл, в котором записаны такие объявления.

### Тело документа

Тело документа состоит из одного или более элементов.

В правильно оформленном XML-документе элементы формируют простое иерархическое дерево, в котором обязательно присутствует **корневой элемент** (root element), в который вложены все остальные элементы документа.

Имена элементов должны быть уникальны в пределах документа. Имя корневого элемента считается именем всего документа и указывается в объявлении типа документа.

<!DOCTYPE root>

### Эпилог

В эпилог XML могут входить комментарии, инструкции по обработке и/или пустое пространство.

В языке XML не определен индикатор конца документа, большинство приложений для этой цели используют завершающий тег корневого элемента документа. Поэтому, встретив завершающий тег корневого элемента, скорее всего, приложение закончит обработку документа, и эпилог обработан не будет.

# Применение стилей CSS

Для отображения XML-документа в браузере необходимо настроить стили каждого из его элементов.

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

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

Ссылка указывается в прологе XML-документа и является инструкцией обработчику.

Настройки корневого элемента аналогичны настройкам тега BODY в HTML и применяются ко всему документу.

В дополнение к ранее изученным настройкам CSS рассмотрим очень важное свойство display.

Это свойство определяет, как элемент должен быть показан в документе. Без него все элементы XML будут отображаться в одну строчку.

Список возможных значений этого свойства, понимаемый разными браузерами очень короткий – block, inline, list-item и none. Все остальные допустимые значения поддерживаются браузерами выборочно.

|  |  |
| --- | --- |
| **Значение** | **Описание** |
| block | Элемент показывается как блочный. Применение этого значения для встроенных элементов заставляет его вести подобно блокам – происходит перенос строк в начале и в конце содержимого. |
| inline | Элемент отображается как часть строки. Значение inline отменяет перенос строк, поэтому содержимое блочных элементов начинается с того места, где окончился предыдущий элемент. |
| list-item | Элемент выводится как блочный и добавляется маркер списка. |
| none | Не отображать элемент в документе. Страница формируется так, словно элемента и не было. Изменить значение и сделать вновь видимым элемент можно с помощью скриптов, обращаясь к свойствам через объектную модель. В этом случае происходит переформатирование данных на странице с учетом вновь добавленного элемента. |
| inline-block | Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу. Фактически такой элемент по своему действию похож на встраиваемые элементы (вроде тега <img>). При этом его внутренняя часть форматируется как блочный элемент, а сам элемент — как встроенный. Не поддерживается версиями IE 6-7. |