**Министерство образования Российской Федерации**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение высшего образования**

**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Р.Е. Алексеева**

**Кафедра «Информатика и системы управления»**

**Управление данными**

**Знакомство с XML**

Методические указания к лабораторной работе №6 для студентов направления 09.03.01«Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии»

**Нижний Новгород**

**2020**

Составитель: Т.И. Балашова

УДК

Базы данных

НГТУ; Сост.: Т.И. Балашова

Н.Новгород 2020. 13 с.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижегородский государственный технический университет.

Типография НГТУ. 603950, Нижний Новгород, ул. Минина, 24.

**Язык разметки XML**

Золотой серединой между языками SGML и HTML стал язык XML (eXtensible Markup Language(расширенный язык разметок)) — это язык разметки документов, первоначально возникший как технология совместного использования документов во Всемирной паутине, но постепенно превратившийся в среду хранения и передачи иерархических структур данных. Это подмножество языка **SGML** (англ. *Standard Generalized Markup Language* — стандартный обобщённый язык разметки) — метаязык, на котором можно определять язык разметки для документов (рис.1). Появился XML в 1986 году.

**GML**

**SGML**

**HTML**

**XML**

**XHTML**

**…...**

Рис. 1. Подмножество языка GML

**Иерархическая модель данных** — это модель данных, в которой данные представляются в виде иерархической (древовидной) структуры. Иерархическая база данных состоит из сущностей, у которых могут быть родительские и дочерние сущности, но обязательно есть одна документная сущность — корневой элемент, с которого все начинается и который не имеет родительского элемента, а все прочие элементы являются его потомками (рис. 2). Иерархические базы данных оптимизированы под чтение информации.

Корневой элемент

Потомок

Потомок

Потомок

Потомок

Потомок

Потомок

Потомок

Потомок

Потомок

Рис. 2. Иерархическая структура данных

XML-документ состоит из сущностей, которые могут содержать символьные данные (англ. character data) и элементы разметки. К разметке относятся теги (англ. tags) — строки текста, заключенные в угловые скобочки (*< >*) и обозначающие границы элементов, а также атрибуты элементов (англ. attributes) и комментарии. Язык XML не содержит набора тегов, подобно HTML, а описывает правила создания своих собственных тегов. Хорошо оформленный документ (*well-formed*) — документ XML, написанный с учетом всех правил.

В начале XML-документа перед корневым элементом располагается пролог. Пролог должен начинаться (для версий языка старше 1.0) с объявления XML, например:

<?xml version="1.0" encoding="UTF-8" ?>

Атрибут (version="1.0") — версию XML, а атрибут (encoding="UTF-8") — кодировку документа.

Специальная инструкция !DOCTYPE, позволяющая с помощью языка схем DTD определить схему документа, т.е. какие элементы и атрибуты у элементов могут быть в данном документе, в каком количестве и возможные способы взаимного «вложения» элементов. Также в прологе могут содержаться инструкции обработки, позволяющие, например, связать XML-документ с таблицами стилей CSS:

<?xml-stylesheet href="style.css"?>

В любом месте XML-документа могут содержаться комментарии, начинающиеся с символов *<!--* и заканчивающиеся символами *-->.*

В одном элементе не может быть двух одинаковых атрибутов (атрибутов с одним и тем же именем). Имена элементов и атрибутов должны начинаться с буквы, символа нижнего подчеркивания ( \_ ) или двоеточия (:). Они могут

содержать только буквы из кодировки Unicode, арабские цифры, дефисы, знаки подчеркивания, точки и двоеточия. Имена не могут начинаться со строки «xml» в любом регистре.

Существуют правила, которые обязательно необходимо соблюдать для правильного построения XML-документов:

1.XML-документ начинается с объявление XML (для версий языка старше 1.0).

2.XML-документ содержит один корневой элемент.

3.В XML-документе комментарии начинаются с символов **<!--** и заканчиваются символами **-->.**

4.В XML-документе имена элементов и имена атрибутов не могут содержать пробелы. Имена элементов регистрозависимые.

5.В XML-документе все открытые теги обязательно должны быть закрыты.

6.В XML-документе не допускается перекрывающихся элементов, то есть необходимо соблюдать уровни вложенности элементов.

В XML-документе в содержимом элемента или значении атрибута могут содержаться специальные символы. Эти символы в некоторых случаях могут быть частью символьных данных, а в некоторых — частями конструкций, входящих в синтаксис XML. Чтобы программы, обрабатывающие XML-документ, правильно понимали, что следует делать со спецсимволами, их необходимо экранировать (рис. 3)

Ссылка на сущность:

& &amp;

< &It;

> &gt;

` &apos;

“ &qout;

Неразрывный пробел –

Ссылка по номеру символа:

& &#38;

< &#60;

> &#62;

` &#39;

“ &#34;

Неразрывный пробел &#160;

Экранирование специальных символов в XML

Рис. 3. Экранирование специальных символов в XML



Рис. 4. Пример составления XML-документа.

Уровни правильности

XML-документа

корректные

валидные

Рис.5. Уровни правильности XML-документа

Валидным (англ. Valid) называется корректный XML-документ, отвечающий дополнительным требованиям к своей внутренней структуре, описанным с помощью языков схем, таких как DTD или XML Schema.

DTD (англ. Document Type Definition — определение типа документа) позволяет задать структуру и допустимый набор элементов в XML-документе и их атрибутов. Если использовать XML-документ как иерархическую базу данных, с помощью DTD можно описать структуру этой базы данных. Приложения могут использовать DTD-схемы для проверки соответствия им (валидации) различных XML-файлов.

DTD-схема может быть объявлена как внутри XML-документа, так и вне его. Для объявления DTD внутри XML-документа используется директива <!DOCTYPE>, которая располагается под объявлением XML. Внутри директивы <!DOCTYPE> указывается корневой элемент XML и описание схемы в квадратных скобках(рис. 6)

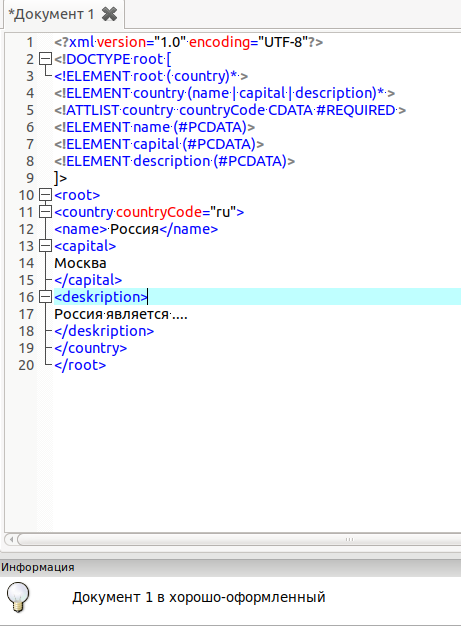


Рис. 5. Пример составления XML документа определение типа документа DTD.

Код DTD-схемы, приведенный на рис.5:

1. !DOCTYPE root — определяет, что корневым элементом данного XML-документа является элемент root;
2. !ELEMENT root(country) — определяет, что элемент root должен содержать вложенный элемент: country;
3. !ELEMENT country(name | capital | deskription) — определяет, что элемент country должен содержать три вложенных элемента: name, capital, deskription;
4. !ATTLIST country countryCode CDATA #REQUIRED — объявление атрибута countryCode, #REQUIRED — указывает на обязательность наличия данного атрибута
5. !ELEMENT name — определяет, что элемент name содержит данные типа #PCDATA;
6. !ELEMENT capital — определяет, что элемент capital содержит данные типа #PCDATA;
7. !ELEMENT deskription — определяет, что элемент deskription содержит данные типа #PCDATA;

При использовании объявления внешнего DTD директива <!DOCTYPE> должна содержать ссылку на DTD-файл, например, через системный идентификатор: <!DOCTYPE root SYSTEM "root.dtd"> . В данном коде: root — название корневого элемента, SYSTEM — указание на то, что DTD-файл располагается на компьютере пользователя, "root.dtd" — название DTD-файл (если указывается только название файла без пути, значит данный файл располагается в той же папке, что и XML-документ).

Ссылка на внешний DTD через публичный идентификатор выглядит следующим образом:

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

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

Сам DTD-файл должен содержать описание схемы, которое при внутреннем объявлении располагается внутри квадратных скобок в директиве <!DOCTYPE>. Объявление элементов в DTD

Для объявления элементов в DTD используется следующий синтаксис:

<!ELEMENT имя\_элемента категория>

или

<!ELEMENT имя\_элемента (содержимое-элемента)>

Пустой элемент объявляется с помощью ключевого слова EMPTY,

указывающего его категорию:

<!ELEMENT имя\_элемента EMPTY>

Некоторые элементы могут содержать только обрабатываемые символьные данные (англ. PCDATA — parsed character data). Эти данные размещаются между начальным и конечным тегами. Эти данные не могут включать символы &, <, или >. Эти символы должны быть экранированы, например, с помощью ссылок на сущности &amp; &lt; и &gt; соответственно.

Если элемент содержит только обрабатываемые символьные данные, то это объявляется указанием ключевого слова #PCDATA внутри скобочек, указывающих на содержимое элемента:

<!ELEMENT имя\_элемента (#PCDATA)>

Элемент с произвольным содержимым объявляется с помощью ключевого слова ANY, указывающего его категорию:

<!ELEMENT имя\_элемента ANY>

Ключевое слово ANY указывает на то, что элемент может содержать как символьные данные, так и любые вложенные элементы. Если элемент содержит дочерние элементы, то их названия объявляются в качестве содержимого элемента внутри круглых скобочек:

<!ELEMENT имя\_элемента (имя\_дочернего\_элемента\_1)>

Или

<!ELEMENT имя\_элемента (имя\_дочернего\_элемента\_1,

имя\_дочернего\_элемента\_2, имя\_дочернего\_элемента\_3)>

Количество дочерних элементов не ограничено. Дочерние элементы, указанные через запятую, должны следовать в родительском элементе именно в том порядке, в котором они указаны в DTD.

Если имя дочернего элемента указано без каких-либо специальных символов в конце, как показано в примере выше, это означает, что в родительском элементе должен содержаться ровно один такой дочерний элемент. Таким образом, код <!ELEMENT root (child1, child2, child3)> означает, что внутри элемента root должно содержаться ровно три дочерних элемента с именами child1, child2 и child3 в указанном порядке.

Существуют следующие специальные символы, позволяющие определить количество возможных вхождений дочернего элемента:

\* — любое количество дочерних элементов (ноль и более);

+ — по крайней мере один дочерний элемент (один и более);

? — необязательное наличие дочернего элемента (ноль или один);

Ниже приведен пример объявления дочернего элемента, который должен входить в родительский как минимум один раз:

<!ELEMENT имя\_элемента (имя\_дочернего\_элемента\_1+)>

Также существует специальный символ | (вертикальная черта), позволяющий задать разделение альтернатив, например:

<!ELEMENT имя\_элемента (доч\_эл\_1| доч\_эл\_2>)

Данный код объявляет, что внутри данного элемента может содержаться либо один элемент доч\_эл\_1, либо один элемент доч\_эл\_2.

Объявления дочерних элементов:

<!ELEMENT root (#PCDATA | name | capital | deskription)\*>

Данный код объявляет, что внутри элемента root могут содержаться вложенные элементы name, capital, deskription, либо обрабатываемые символьные данные, причем в любом порядке и количестве.

Объявление атрибутов в DTD

Для объявления атрибутов используется следующий синтаксис:

<!ATTLIST имя\_элемента имя\_атрибута тип\_атрибута свойство\_атрибута>

Недостатки DTD:

* В DTD используется отличный от XML синтаксис языка, что может приводить к различным ошибкам, например, проблемам с кодировкой.
* В DTD нет проверки типов данных (строка, целое число, число с плавающей точкой, логический тип данных, дата и время и т.д.), используется только один тип данных — строка.
* В DTD нет пространств имен, из-за чего нельзя поставить в соответствие XML-документу две и более DTD-схемы.

Для устранения этих недостатков был разработан язык XML Schema

**Часть 1**

Задание для первой части.

Составьте свой корректный XML-документ, описывающий любую иерархическую базу данных. Документ должен содержать несколько различных типов вложенных элементов и не менее четырех уровней вложенности. Элементы должны содержать различные атрибуты и текстовые

значения. Используя схему базы данных из первой лабораторной работы для составления XML-документа.

Отчет должен содержать: листинг XML-документа, описание элементов, вывод о проделанной работе.

Контрольные вопросы

1. Что такое XML?
2. Что такое иерархическая модель данных?
3. Что могут содержать сущности, из которых состоит XML-документ?
4. Для чего в XML-документах используются теги?
5. Что идет перед корневым элементом XML-документа?
6. Для чего в XML-документах используются инструкции обработки?
7. Что такое начальный и конечный теги?
8. Что может быть именем элемента или атрибута в XML-документе?
9. Назовите правила для построения корректных XML-документов.
10. Как в XML-документе обозначаются комментарии?
11. Допустимы ли в XML-документе не закрытые теги?
12. Что такое спецсимволы в XML-документах?
13. Как можно экранировать спецсимволы в XML-документах?
14. Что такое DTD?
15. Что такое валидация XML-документа?
16. С помощью какой директивы объявляется DTD внутри XML-

документа?

1. Какие недостатки языка DTD существуют?

**Часть 2**

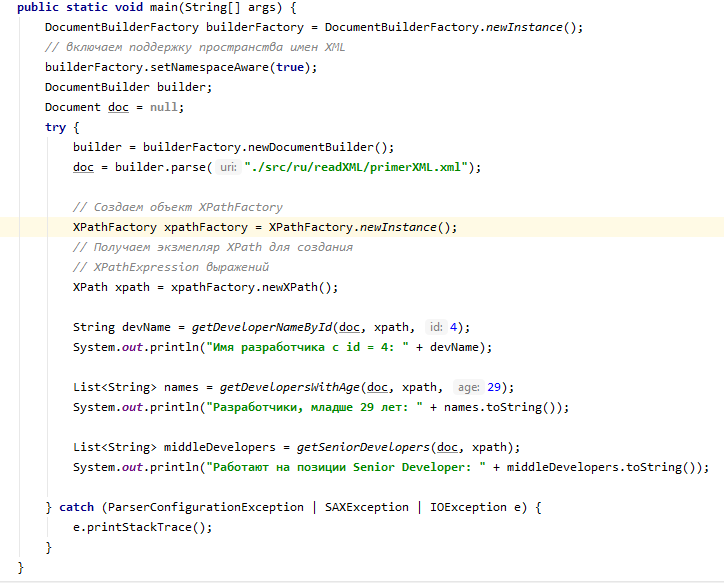
**Теоретическая часть**

Язык запросов XML используется для доступа к элементам XML-документа. Результатом вычисления XPath-выражения (запроса) является указание на определенный участок или участки документа. XML-документ представляет собой дерево, корнем которого является корневой элемент. От корня отходят ветви, заканчивающиеся узлами (англ. nodes) — элементами, их атрибутами и текстовым содержимым. У каждого узла нижнего уровня может быть только один прямой предок, но при этом потомками одного элемента может быть множество элементов.

Узел атрибута или текстового содержимого элемента не может иметь потомков.

Язык Xpath позволяет различать следующие типы узлов: корень (document node), элемент (element node), атрибут (attribute node), текст (text node) и комментарий (comment node). Корневой узел содержит не только корневой элемент документа, но и все прочие узлы, в том числе инструкции обработки, пространства имен, комментарии и текстовые данные. Корневой узел является контейнером для остальных узлов. Имена корневого узла и корневого элемента документа совпадают. Также совпадают имена узлов-элементов и узлов-атрибутов с именами соответствующих элементов и атрибутов. Узлы-комментарии заносятся в дерево без имени. Также не имеют имени текстовые узлы — они содержат текст, находящийся в XML-документе между элементами.

Рис.6. Пример создания запроса к XML документу.



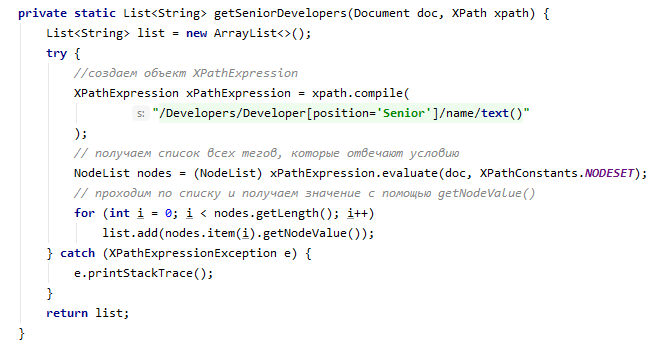


Рис.7. Пример получение конкретной информации из XML документа.



Рис.8. XML документ к которому был написан запрос.

Задание для второй части.

Ознакомится с языком запросов XPath. Написать пару запросов к своему XML документу созданному ранее.

Отчет должен содержать: листинг программы, результат вывода программы, вывод о проделанной работе.